UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Uroš Kos

Sistem za daljinsko merjenje nivoja vode v zbiralniku

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2016

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Uroš Kos

Sistem za daljinsko merjenje nivoja vode v zbiralniku

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: mag. Igor Škraba

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V nalogi razvijte strojno in programsko opremo za merjenje nivoja vode v oddaljenem zbiralniku. Sistem naj omogoča periodično merjenje nivoja, brezžični prenos rezultatov meritev na oddaljeni strežnik pri porabniku vode, hranjenje podatkov na strežniku in prikaz podatkov na mobilni napravi. V primeru kritičnega nivoja vode naj sistem proži opozorila po elektronski pošti.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Uroš Kos, vpisna številka 63010346, avtor zaključnega dela z naslovom:

*Sistem za daljinsko merjenje nivoja vode v zbiralniku (*angl. *System for remote measuring of water level in tank)*

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno pod mentorstvom pred. mag. Igorja Škraba;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil/-a vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil/-a;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil/-a soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 15. julija 2016 Podpis študenta/-ke:

Zahvaljujem se družini za vso pomoč in podporo pri študiju. Posebej se zahvaljujem mentorju, pred. mag. Igorju Škrabi za nasvete in pomoč pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
SOC	system on chip	računalnik na enem čipu
RAM	random access memory	bralno-pisalni pomnilnik
CMS	content management system	sistem za upravljanje vsebin
HDMI	high definition multimedia interface	visokoločljivostni multimedijski vmesnik
GSM	global system for mobile communications	globalni sistem za mobilne komunikacije
LAN	local area network	lokalno omrežje
PNP	plug and play	vklopi in uporabljaj
IDE	integrated development environment	integrirano razvojno okolje
USB	universal serial bus	univerzalno serijsko vodilo
DIP	dual in-line package	vrsta ohišja integriranih vezij
PWM	pulse width modulation	pulzno širinska modulacija
ADC	analog to digital converter	analogno-digitalni pretvornik
AC	alternating current	izmenični tok
DC	direct current	enosmerni tok
RISC	reduced instruction set computer	procesor z omejenim naborom ukazov
USART	universal synchronous asynchronous receiver transmitter	univerzalni sinhronski asinhronski sprejemnik in oddajnik
UART	universal asynchronous receiver transmitter	univerzalni asinhronski sprejemnik in oddajnik
SPI	serial peripheral interface	serijski periferni vmesnik

I2C	two wire interface	vrsta vodila
MIPS	millions of instructions per second	milijon ukazov na sekundo
IEEE	Institute of electrical and electronics engineers	Inštitut inženirjev elektrotehnike in elektronike
GPRS	general packet radio service	paketni prenos podatkov
SMS	short message service	sistem kratkih sporočil
IR	infra red	infrardeča svetloba
RF	radio frequency	radijska frekvenca
RPSMA	reverse polarity subminiature version A	konektor mini A z obrnjeno polariteto
SIM	subscriber identity module	mobilna kartica za prepoznavo
AVR	Alf Vegard RISC	oznaka za arhitekturo mikrokrmilnika
ISM	industrial scientific and medical	industrijski, znanstveni in medicinski
FTDI	Future technology devices international	podjetje, specializirano za povezave starejših serijskih standardov na USB
LCD	liquid crystal display	zaslon s tekočimi kristali
COM	communication port	komunikacijski vmesnik
GUI	graphical user interface	grafični uporabniški vmesnik
HTML	hyper text markup language	označevalni jezik
CSS	cascading style sheets	kaskadne stilske predloge
MVC	model – view – controller	model – pogled – krmilnik
SQL	structured query language	strukturiran povpraševalni jezik
TCP	transmission control protocol	protokol za nadzor prenosa
RTC	real time clock	ura realnega časa
AJAX	asynchronous JavaScript and XML	asinhroni JavaScript in XML

Povzetek

Naslov: Sistem za daljinsko merjenje nivoja vode v zbiralniku

Tematika naloge je izdelati celovit sistem za merjenje nivoja vode v zbiralniku (vodnem zajetju) na oddaljeni lokaciji ter prikazati podatke na mobilni napravi. Naloga zajema pripravo strojne in programske opreme za meritve nivoja ter pripravo in implementacijo strežnika, ki hrani periodično posredovane podatke v svoji podatkovni bazi. Zbrani podatki so dostopni različnim odjemalcem preko mobilne spletne aplikacije. Odjemalec ima vpogled v trenutno stanje in možnost pregleda zgodovine po obdobju (trend nihanja). V primeru mejnih nihanj, ko je na primer dosežena minimalna vrednost, lahko strežnik proži opozorila preko elektronske pošte.

Ključne besede: AVR, ATmega328, mikrokrmilnik, Arduino, ZigBee.

Abstract

Title: System for remote measuring of water level in tank

In this thesis we created a comprehensive system for measuring water level in tank at a remote location and display the data on a mobile device. The tasks cover the preparation of hardware and software for level measurement, as well as the preparation and implementation of the server that holds data in its database. Data is send to server periodically from hardware unit and is available to different clients via mobile web application. The client has access to the current status and the possibility of reviewing the history (trend fluctuations). In the case of predefined thresholds, for example reached the minimum value, the server triggers alerts via e-mail.

Keywords: AVR, ATmega328, microcontroller, Arduino, ZigBee.

Poglavje 1 Uvod

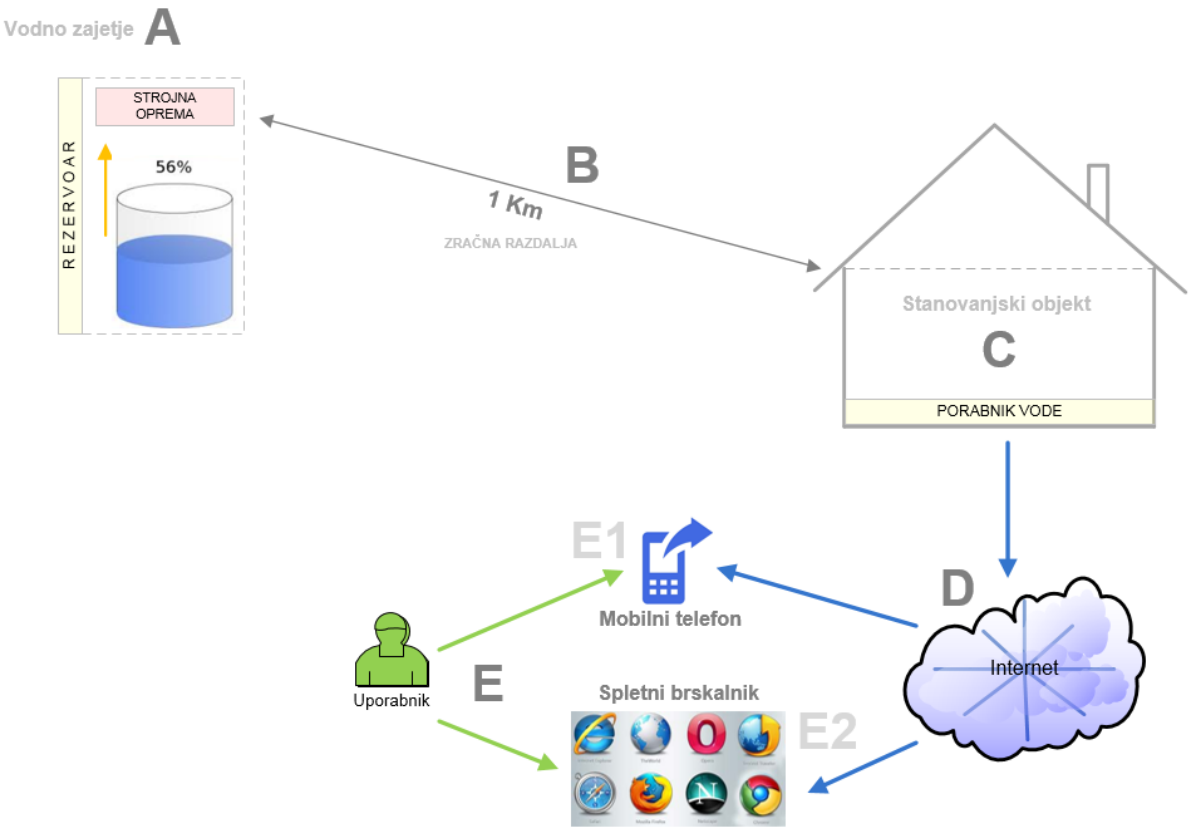
V diplomskem delu obravnavam in vzpostavljam rešitev za daljinsko merjenje nivoja vode v oddaljenem zbiralniku. Rešujem povsem praktično težavo, saj je zbiralnik dislociran od stanovanjskega objekta, ki ga uporablja, hkrati pa je močno odvisen od njega. Razmere se med letom spreminjajo in nemalokrat se zgodi, da vode v vodnem zajetju ni zadosti in je potrebna prilagoditev porabe. Razumljivo je to lažje, če so porabniki pravočasno obveščeni. Popolno pomanjkanje vode lahko povzroči tudi težave, tako da je končni cilj vsekakor, da sistem sam proži obvestila, še posebno v mejnih primerih oziroma da je za končnega uporabnika storitev transparentna.

V okviru te naloge tako vzpostavljam napravo za merjenje količine oziroma višine vode, ki je avtonomna in primerna za zunanje delovanje. Opisan bo postopek izbire mikrokrmilnika in senzorjev glede na zahteve, v nadaljevanju pa tudi strežniškega sistema ter medsebojne povezave teh naprav. Za konec bo pripravljena enostavna mobilna spletna aplikacija, ki bo omogočala vpogled v podatke in ponastavitev parametrov naprave.

V diplomskem delu bom uporabljal več različnih tehnologij, vključno s strojno opremo, vse to s ciljem, da se te povežejo v končni produkt, ki bo uporaben v praksi.

1.1 Idejni načrt sistema za daljinsko merjenje nivoja

Na sliki 1 je predstavljen idejni načrt sistema, iz katerega so razvidne razmere na terenu oziroma obseg sistema. Črke nakazujejo posamezne segmente: fizično mesto, napravo, način prenosa podatkov ali medij. Če je mogočih več rešitev, se posamezni segment še razdeli na opcijo 1 in 2. Tako imamo na primer nakazanih več možnosti za prenos podatkov do končnega uporabnika.



Slika 1 – Idejni načrt sistema za daljinsko merjenje nivoja vode

Pomen označb na sliki:

- A – vodno zajetje oziroma v nadaljevanju zbiralnik;
- B – zračna razdalja med zbiralnikom in stanovanjskim objektom;
- C – stanovanjski objekt in tudi porabnik vode iz zbiralnika, katerega nivo vode želimo meriti; v objektu je tudi naš center, kjer želimo podatke hraniti;
- D – internet (medij, s katerim želimo, da so podatki na voljo);
- E – način prikaza podatkov končnemu uporabniku. Končni uporabnik lahko podatke spremlja v stanovanjskem objektu, to je na domačem namiznem računalniku, ali daljinsko, na primer na mobilni napravi. Zahteva in želja sta, da lahko podatke spremljamo, tudi če nismo doma, in enako prejemamo morebitna opozorila.

1.2 Koraki za vzpostavitev sistema

Naloga zajema vse zgoraj omenjene segmente, nekoliko večji poudarek je na strojni in programski opremi naprave za merjenje nivoja vode v zbiralniku, prenosu podatkov do stanovanjskega objekta in vzpostavitvi strežnika za hranjenje in posredovanje podatkov (A, B in C na sliki 1). Sistem gradimo postopoma in tako v zbiralniku najprej pripravimo strojno opremo za merjenje nivoja in posredovanje podatkov v center. Osnova strojne opreme bo gotovo mikrokrmilnik, ki bo nadzoroval merjenje ter javljal podatke v center in prejemal morebitne zahtevke iz centra. Nato je potrebno zagotoviti mesto za hranjenje podatkov – strežnik. Sledi vzpostavitev povezave med mikrokrmilnikom in strežnikom. Potem sledi drugi del, kjer zajete podatke pripeljemo do uporabnika (D in E na sliki 1).

V nalogi so podrobneje opisani naslednji ključni koraki:

- izbira mikrokrmilnika,
- izbira senzorja za opravljanje meritev,
- priprava strežnika za sprejem podatkov in hranjenje,
- vzpostavitev dvosmerne komunikacije med mikrokrmilnikom in strežnikom,
- priprava mobilne spletne aplikacije za prikaz podatkov.

Poglavje 2 Mikrokrmilnik

2.1 Splošno o mikrokrmilnikih

Mikrokrmilnik je čip, ki deluje kot mikroračunalnik (SoC) in ima vse potrebne elemente za delovanje že vgrajene v integrirano vezje. Mikrokrmilniki so posebej namenjeni zaključenim aplikacijam ali uporabi v vgrajenih sistemih (angl. *embedded applications*, *embedded systems*) prav zaradi svoje majhnosti in enostavnosti vezja.

Tipični mikrokrmilnik vsebuje CPE, pomnilnik in programabilni V/I-sistem (angl. *I/O ports*). Centralna procesna enota (CPE) upravlja delovanje mikrokrmilnika na podlagi programa, ki je shranjen v programskem pomnilniku. Podatkovni pomnilnik (RAM) hrani podatke, ki jih CPE obdeluje, oziroma vrednosti uporabniških spremenljivk. Periferni vmesniki (V/I-sistem) omogočajo povezavo z okolico.

Prav programabilnost oziroma možnost relativno enostavnega programiranja je razlog za velik porast mikrokrmilnikov in dejstvo, da jih danes najdemo v večini modernih elektronskih naprav. Pred mikrokrmilniki se je uporabljala diskretna logika, ki pa zahteva več komponent, več testiranja in ni prilagodljiva. Mikrokrmilniki so cenejši, razvoj je hitrejši, predvsem pa so se sposobni hitro prilagajati spremembam. Če želimo spremeniti delovanje, v idealnem primeru samo zamenjamo program, celotno vezje pa ostane nespremenjeno.

2.2 Izbira mikrokrmilnika

Izbira mikrokrmilnika je prva naloga, na katero naletimo pri projektu. Mikrokrmilnik bo srce strojne opreme za merjenje in za pošiljanje podatkov v centralni strežnik. Hkrati bo zadolžen tudi za sprejem morebitnih ukazov iz centra ter ponastavljanje režima delovanja.

Izhodiščne zahteve za mikrokrmilnik so:

- enostavna uporaba v razvojnem in produkcijskem okolju,
- enostaven priklop in kompatibilnost z različnimi senzorji ali drugimi enotami,
- majhna poraba energije,

- enostavno programiranje in razhroščevanje.

Dodatne tehnične zahteve:

- delovanje v temperaturnem območju od –10 ºC do +40 ºC,
- možnost priklopa na usmernik (220 V) in možnost avtonomnega delovanja (baterijsko napajanje po potrebi).

2.3 Mikrokrmilniška razvojna orodja

Leta nazaj je bila uporaba mikrokrmilnikov predvsem v domeni usposobljenih elektrotehnikov/elektronikov ali drugih izkušenih poznavalcev, ki so imeli specifična znanja in opremo za programiranje tovrstnih enot. Zadnja leta so mikrokrmilniki vse bolj popularni tudi med ljubiteljskimi razvijalci. Tako je področje postalo zelo dostopno različnim zanesenjakom, ki želijo enostavno in hitro najti rešitve za prosti čas kot tudi za bolj resne projekte na področjih domače/hišne avtomatizacije (na primer pametne hiše), avtomatizacije naprav in strojev (na primer namakalni sistemi), telemetrije, modelarstva, centralnih sistemov in tako dalje. Mikrokrmilniki se uporabljajo tudi v robotiki, ki je še posebno v porastu v zadnjih letih.

V ta namen so se mikrokrmilniška okolja zelo približala končnemu uporabniku, ki ne potrebuje več nujno točno določenega in poglobljenega strokovnega znanja ali drage opreme. Okolje lahko že vključuje vse potrebne komponente in pribor (angl. *development kit*), kar omogoča lažji začetek dela in vzpostavitev začetnega razvojnega cikla.

2.3.1 Odprtokodne strojno-razvojne platforme

Zanimiv je pojav odprtokodnih strojnih platform, ki v marsičem spominjajo na zelo poznane odprtokodne programske rešitve, na primer s področja spletnih tehnologij. Danes večine spletnih mest ne razvijamo več od začetka, temveč so popularni odprtokodni sistemi CMS (angl. *Content Management System*), ki omogočajo veliko hitrejšo gradnjo spletnih rešitev, prinašajo že vgrajene funkcionalnosti in hkrati slonijo na že preverjeni osnovi (angl. *framework*). Tipični predstavniki odprtokodnih sistemov CMS so Joomla, Wordpress, Drupal, Magento, Typo in tako dalje. S takim sistemom (arhitekturne zasnove) lahko osnovno spletno stran postavimo v nekaj urah ali celo minutah.

Odprtokodne strojno-razvojne platforme enako ponujajo odlično izhodišče za razvoj prototipnih in povsem komercialnih produktov. Običajno ponujajo predvsem širok spekter že

pripravljenih projektnih rešitev, modularnost in ne nazadnje podporo široke skupnosti uporabnikov.

Tipične razvojne platforme s področja elektrotehnike:

- Arduino (2004 Italija),
- Raspberry Pi (2012 Anglija, OS: Linux),
- Netduino (večopravilnost, podpora za .NET framework),
- BeagleBoard (2008, OS: Linux),
- NodeMCU (2014),
- OpenRISC.

2.3.2 Izbira strojne platforme

Zahteve za izbiro strojne platforme so naslednje:

- **Enostaven razvoj programa in programiranje**

  Nujno je, da za programiranje ne potrebujemo dodatne strojne opreme, kot je na primer ločen programator. Če ima platforma svoje razvojno okolje, je to dodatna prednost.

- **Široka skupnost uporabnikov**

  Močna in aktivna skupnost uporabnikov pomeni, da gre za živo in prepoznavno platformo, ki ima tudi veliko dokumentiranih uporabniških rešitev. Med uporabniki je mogoča tudi hitra izmenjava mnenj in izkušenj.

- **Modularni dodatki in razširitve**

  Zaželeno je, da platforma omogoča enostavno dodajanje preverjenih razširitvenih modulov in senzorjev. Uporaba teh mora biti ustrezno dokumentirana.

- **Primerno za enostavne rešitve**

  Okolje mora biti sorazmerno težavi, ki jo rešujemo.

Platforma Arduino je posebno primerna za nizkocenovne rešitve in samostojne aplikacije z majhno porabo energije. Izvedenka Netduino je primerna za kompleksne mrežne rešitve, še

posebno pri povezovanju z drugimi Microsoftovimi tehnologijami. RasperyPi in BeagleBone delujeta na operacijskem sistemu Linux, vključujeta tudi periferne enote, kot sta HDMI in Ethernet, in sta tako računalnik v malem. RasperyPi je primeren za grafično intenzivne aplikacije, medtem ko ima BeagleBone izredno zmogljivo strojno opremo in zato tudi višjo ceno.

Na podlagi zgornjih kriterijev in kriterijev, ki se nanašajo neposredno na mikrokrmilnik, smo izbrali platformo Arduino. Platforma Arduino je bila izbrana tudi zaradi ogromne skupnosti in veliko različnih dodatnih komponent, kot so vhodna tipala in razširitveni moduli (angl. *shields*), ki se na ploščo Arduino samo pritrdijo in že ponujajo dodatno funkcionalnost. Taki moduli so na primer modul GSM, modul WiFi, modul LAN in tako dalje.

2.4 Platforma Arduino

Arduino je odprtokodna platforma [1] za izdelavo projektov s področja elektronike. Arduino zajema strojno opremo (angl. *board*) in programsko okolje za programiranje ter nalaganje programa. Arduino popolnoma sledi principu vstavi-in-poženi (angl. *PnP*).

Platforma Arduino s seboj prinaša programsko okolje Arduino (Arduino IDE), ki omogoča enostavno pripravo programa, podporo za določene razširitvene knjižnice, ki so že del okolja, in opcijo za prenos programa na čip.

Platforma je uporabna in priljubljena tako med naprednimi uporabniki zabavne elektronike kot začetniki. Razlog za privlačnost je v preprosti uporabi, kjer ni potrebe po dodatnem programatorju za mikrokrmilnik. Ploščo Arduino preprosto povežemo z računalnikom preko vmesnika USB in že je pripravljena za testiranje in reprogramiranje. Ko je testiranje zaključeno, lahko tako ploščo odklopimo od računalnika in jo uporabljamo neodvisno z lastnim napajanjem. Dodatna in naprednejša možnost je, da na plošči odstranimo samo čip (mikrokontroler odstranimo iz podnožja pri DIP-ohišju) in ga uporabimo v svojem vezju. V poenostavljenem in prilagojenem vezju (brez dodatnih komponent, na primer napetostnega regulatorja) porabo toka tako še zmanjšamo, kar je zlasti primerno, ko napajanje zagotavlja baterija ali je potrebna minimalna velikost sistema.

Število uporabnikov platforme po nekaterih podatkih še vedno skokovito raste, hkrati pa se Arduino kot telematska platforma uspešno uporablja tudi v pedagoškem procesu [2].

2.4.1 Izbira ustrezne izvedbe

Izbira ustrezne izvedbe Arduino (plošče) zahteva najprej pregled vseh možnosti. Skupnost Arduino ponuja več strojnih izvedenk, ki se razlikujejo po uporabljenem mikrokrmilniku Atmel, številu serijskih vrat, vhodov A/D, izhodov PWM, pomnilniku in še nekaterih drugih karakteristikah.

Tabela 1 – Primerjava različnih izvedb platforme Arduino

Model	CPE	Frekv.	Analogni V/I	Digitalni VI/PWM	EEPROM [KB]	SRAM [KB]	FLASH [KB]
UNO	ATMega328	16 MHz	6/0	14/6	1	2	32
MEGA	ATmega2560	16 MHz	16/0	54/15	4	8	256
Gemma	ATtiny85	8 MHz	1/0	3/2	0.5	0.5	8
Nano	ATmega328P	16 MHz	8/0	14/6	1	2	32
LilyPad	ATMega328	8 MHz	4/0	9/4	1	2	32

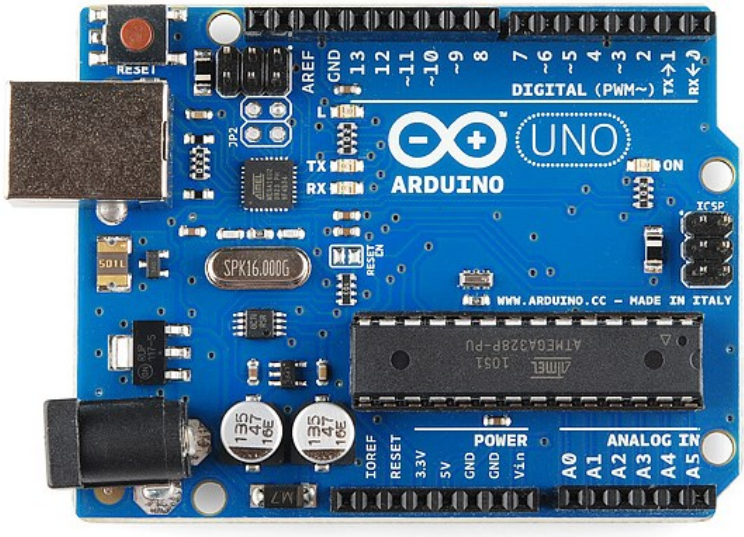
Več podatkov in celoten nabor različnih izvedb platforme Arduino je dostopnih na domači strani [3].

Izbrali smo izvedbo Arduino UNO zaradi naslednjih značilnosti:

- Plošča ima vmesnik USB in s tem povezavo do okolja Arduino IDE za programiranje ter razhroščevanje. USB-povezava se med priklopom uporablja tudi kot napajanje (5 V).
- Mikrokrmilnik na plošči je v ohišju DIP[1] oziroma DIP28 (angl. *dual-in-line package*), kar je primerno za enostavno menjavo mikrokrmilnika ali samostojno postavitev.
- Plošča ima pripravljen regulator napetosti za zunanje 9V-napajanje (AC/DC-usmernik) ali 9V-baterijo, kar povežemo na pripravljen 2,1-milimetrski vtič.
- Mikrokrmilnik na plošči ima 14 digitalnih vhodov/izhodov in 6 analognih vhodov, kar je povsem dovolj za trenutni obseg in hkrati pušča precejšnje možnosti za razširitev.

[1] Dual in line package (DIP) je čip pravokotne oblike, ki ima dve vzporedni liniji pinov. Tak čip je zelo priročen za uporabo na razvojni plošči (angl. *protoboard*).

- Arduino UNO (in predhodno Arduino Duemilanove) je ena najbolj razširjenih izvedenk, ponuja odlično razmerje med ceno ter zmogljivostjo in tudi omogoča neposredne nadgradnje z zelo popularnimi razširitvenimi moduli Arduino.



Slika 2 – Mikrokrmilniška plošča Arduino UNO

2.4.2 Specifikacije mikrokrmilnika ATmega328P

Na plošči Arduino UNO je uporabljen mikrokrmilnik Atmel Atmega328. To je 8-bitni AVR[2] RISC-mikrokrmilnik, ki deluje s frekvenco ure 16 MHz. Mikrokrmilnik AVR ima harvardsko arhitekturo pomnilnika, to je ločen pomnilnik za ukaze in operande.

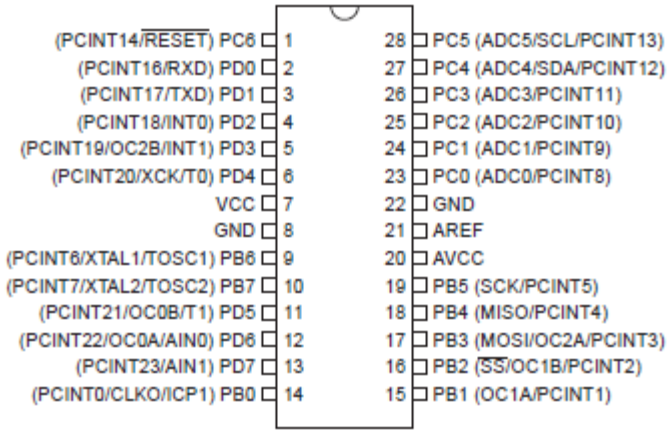
Mikrokrmilnik ima 6 analognih vhodov in 14 digitalnih vhodov/izhodov (V/I), od teh jih lahko 6 uporabimo kot PWM-izhode. PWM (angl. *Pulse Width Modulation*) oziroma pulzno-širinska modulacija je modulacijska tehnika, ki se uporablja pretežno za krmiljenje moči na električnih napravah. Analogni vhodi imajo 10-bitno resolucijo, digitalni izhodi PWM pa 8-bitne izhode.

Mikrokrmilnik ima tri časovnike ter možnost uporabe internih in zunanjih prekinitev. Tovrstne prekinitve lahko uporabimo, če na primer postavimo mikrokrmilnik v enega od načinov varčevanja (podprtih je 5 nivojev/stanj), pri čemer se mikrokrmilnik prebudi, ko preteče čas ali pride do zunanje prekinitve.

[2] AVR-arhitekturo sta prvotno razvila in predstavila dva študenta v okviru diplomskega dela na inštitutu za tehnologijo na Norveškem (NTH – Norwegian Institute of Technology). Tehnologijo je kasneje kupil in dokončno razvil Atmel. Leta 1996 je bil AVR eden prvih v družini mikrokrmilnikov, ki je uporabljal FLASH-pomnilnik za hranjenje programske kode za razliko od krmilnikov, ki so uporabljali ROM, EPROM ali EEPROM.

Dodatno ima krmilnik tudi programirljiv serijski vmesnik USART, serijski vmesnik SPI, programirljiv časovnik Watchdog z ločenim urinim signalom in možnost komunikacije po protokolu I^2C (angl. *2-wire serial interface*).

Družina mikrokmilnikov AVR je izredno popularna, razlog je gotovo v možnosti programiranja v programskem jeziku C, kar precej olajša programiranje v primerjavi s programiranjem v zbirnem jeziku. Pri izvajanju ukazov dosega mikrokrmilnik približno 1 MIPS na MHz.



Slika 3 – Signali mikrokrmilnika ATmega328P v DIP-ohišju

Na sliki 3 je prikaz signalov mikrokrmilnika ATmega328P. V tabeli 2 so ključne tehnične karakteristike, več pa v spletni dokumentaciji [4]. Če je čip na plošči Arduino, ima že vnaprej naložen zagonski nalagalnik oziroma bootloader[3].

Tabela 2 – Tehnične karakteristike ATmega328P

Napajalna napetost	3 V–5 V
Digitalni V/I	14 (pri čemer 6 omogoča PWM-način delovanja)
Analogni V/I	6
PWM V/I	6
Flash-pomnilnik	32 KB (ATmega328P)
SRAM	2 KB
EEPROM	1 KB

[3] Bootloader ali zagonski nalagalnik je program, ki steče ob vsakem resetu in naloži po potrebi nov program v programski pomnilnik. Bootloader Arduino omogoča enostavno programiranje čipa preko vmesnika USB.

Frekvenca ure	16 MHz
Dolžina	68,6 mm
Širina	53,4 mm
Teža	25 g

2.4.3 Vhodne in izhodne naprave Arduino

2.4.3.1 Senzorji

Na Arduino lahko priklopimo ogromno število dodatnih senzorjev ali tipal in v tem je tudi privlačnost platforme. Direkten priklop je mogoč za vse elemente, ki imajo napetost v območju 3,3 V–5 V. Za komunikacijo lahko uporabimo serijsko povezavo 1-Wire (MicroLan) ali vodilo I^2C.

Povezava 1-Wire je izredno popularna pri enostavnih elementih, kot so na primer temperaturni senzorji, pri katerih odčitavamo eno vrednost in so hitrosti nizke. Vodilo I^2C omogoča vzporedno povezavo več senzorjev ali naprav na mikrokrmilnik pri nizkih hitrostih.

2.4.3.2 Razširitveni moduli

Razširitveni moduli Arduino razširjajo osnovno funkcionalnost plošče z dodatnimi V/I-enotami. Za tovrstne module je značilno, da uporabljajo natično letvico in se na ploščo Arduino samo nataknejo brez dodatnega ožičenja. Moduli se lahko nalagajo v višino eden na drugega. Primeri tipičnih razširitvenih modulov so modul Ethernet, modul Wi-Fi, modul GPRS/GSM, modul CAN-BUS, modul XBee in drugi.

2.4.4 Arduino IDE

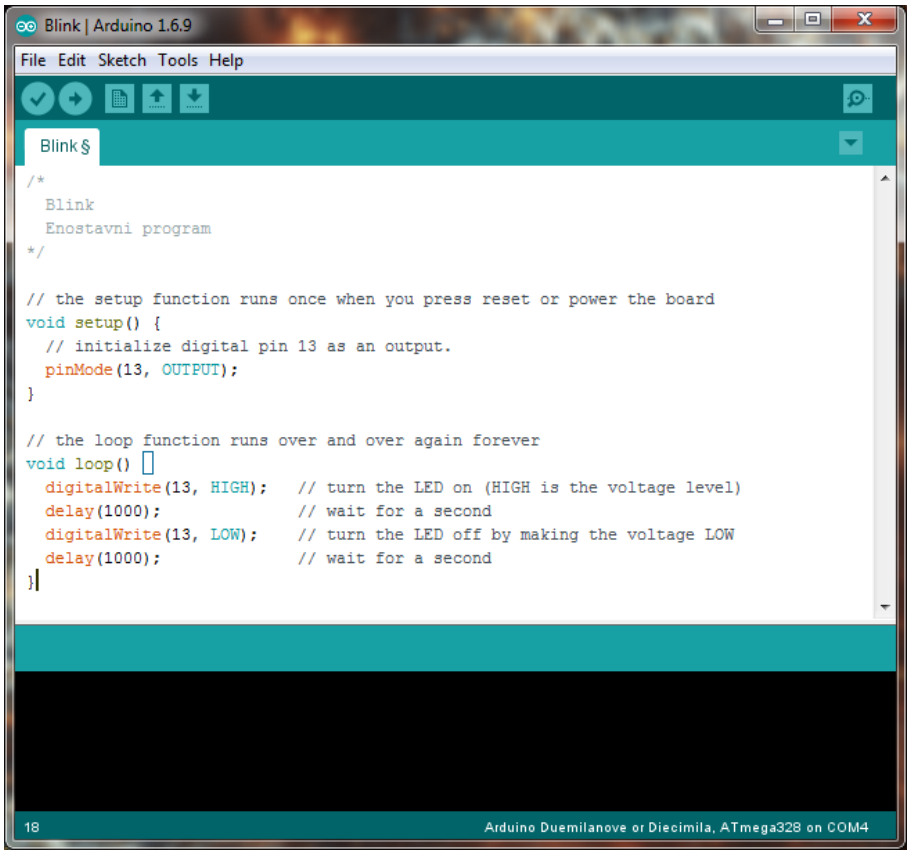
Odprtokodna programska oprema Arduino (IDE – angl. *Integrated Development Enviroment*) omogoča enostavno pisanje, prevajanje in prenos programa na poljubno ploščo Arduino. Okolje je pisano v Javi in podpira sisteme Windows, Max OS X in Linux.

Program, napisan v okolju Arduino, se imenuje »sketch« in ima značilno končnico .ino. Osnovni program je vedno v eni datoteki, ki lahko vključuje več podpornih knjižnic. Tipično za programski jezik C jih dodamo s stavkom »include«. Pred vsakim nalaganjem programa na ploščo Arduino se izvrši verifikacija, ki pri sintaktičnih napakah oziroma napakah pri prevajanju postopek zaustavi. Pri uspešno prevedenem in naloženem programu dobimo še zelo uporabno informacijo o velikosti programa v pomnilniku.

Okolje pozna poleg standardnih ukazov, ki se uporabljajo pri jeziku C, še nekaj prilagojenih ukazov posebno za Arduino, kot so pinMode(), analogRead(), digitalWrite().

2.4.4.1 Primer enostavnega programa

Primer enostavnega programa in hkrati tudi okolja Arduino IDE je na sliki 4.



Slika 4 – Primer enostavnega programa v Arduino IDE

Osnovni metodi, ki ju okolje Arduino že ponudi pri pripravi novega programa, sta:

- setup() – metoda se požene samo na začetku in je primerna za prvo inicializacijo,
- loop() – metoda je v obliki neskončne zanke.

Vse globalne spremenljivke lahko definiramo na začetku (na primer pred metodo setup) in so tako dostopne v vseh metodah, ki sledijo. V zgornjem primeru imamo samo osnovno metodo »loop«, znotraj katere nastopa vsa koda. V realnem in zahtevnejšem primeru bi bila – in kasneje tudi bo – koda razdeljena na več metod.

Poglavje 3 Meritev nivoja vode v zbiralniku

3.1 Merjenje nivoja vode na osnovi razdalje

Velikost zbiralnika, katerega nivo vode želimo izmeriti, je 1,5 m x 1 m x 1 m (dolžina x širina x višina). Ker želimo meriti relativno količino oziroma napolnjenost zajetja, je tako dovolj, da merimo, na kateri višini je trenutno gladina vode. V ta namen bi lahko na primer uporabili princip mehaničnega plovca. Takoj v naslednjem koraku pa sledi ključni moment pri meritvi, kako vhodni podatek enostavno pretvoriti v ustrezno digitalno ali analogno vrednost, ki jo lahko prebere mikrokrmilnik na svojem vhodu.

Izhodišče je, da uporabimo senzorje ali tipala, ki so kompatibilni s platformo Arduino oziroma čipom ATmega328P. Zaželeno je tudi, da je senzor mogoče priklopiti brez dodatnega vezja.

Široko ponudbo senzorjev lahko izberemo iz spletne trgovine, kot je na primer Sparkfun [5]. Senzor za direktno merjenje višine vode ni na voljo, lahko pa to nalogo precej uspešno opravlja senzor za razdaljo, ki meri višino gladine vode posredno, tako da meri razdaljo od zgornjega roba (pokrova) do gladine.

Pri odčitavanju natančnost meritve ni vitalnega pomena in je dopustna napaka tudi kakšen centimeter ali dva. Za končnega uporabnika je pomembna informacija, izražena v odstotkih napolnjenosti zbiralnika, na primer 40 odstotkov, 60 odstotkov, 80 odstotkov in tako dalje.

Enostavni in popularni senzorji razdalje so dveh vrst:

a) IR-senzor

Infrardeči senzor oddaja IR-svetlobo in meri kot, pod katerim ta snop pride nazaj. IR-senzor zaradi svetlobe, ki jo oddaja, ne deluje najbolje na soncu. Deluje pa precej zanesljivo na dnevni svetlobi.

b) Ultrazvočni senzor

Ultrazvočni senzor uporablja zvok namesto svetlobe in bi mu lahko rekli tudi sonar. Prisotnost sončne svetlobe ga ne moti, lahko ga moti zvočni odboj oziroma odmev, če je prostor tak, da odmev nastane.

3.2 IR-Senzor Sharp GP2D12

Za senzor razdalje smo izbrali infrardeči senzor Sharp GP2D12:

- senzor ima ustrezno merilno območje (10 do 80 cm),
- senzor ima ustrezno temperaturno območje,
- senzor vrača rezultat kot analogno vrednost na enem signalnem vodniku.

Senzor Sharp GP2D12 na sliki 5 deluje na osnovi triangulacije. Senzor odda svetlobni žarek (valovna dolžina 850 nm +/– 70 nm) v smeri ovire, procesna enota senzorja pa meri kot, pod katerim se svetloba vrne ali se sploh ne vrne. Senzor rezultat vrača v obliki napetosti na analognem izhodu v območju od 0,45 V do 2,45 V, kar ustreza razdalji od 10 cm do 80 cm.

Slika 5 – Senzor Sharp GP2D12

3.2.1 Specifikacije senzorja

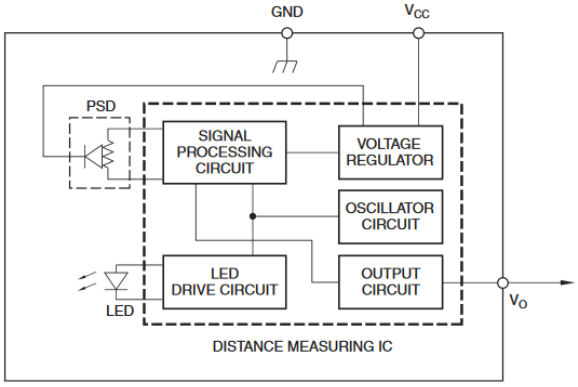
Natančno karakteristiko senzorja je mogoče videti v tehnični specifikaciji [6]. V tabeli 3 so glavne tehnične karakteristike.

Tabela 3 – Tehnične karakteristike Sharp GP2D12

Merjena razdalja	10 do 80 cm
Delovna napetost	4,5 V–5,5 V
Poraba toka	33 mA
Delovna temperatura	–10 °C do 60 °C
Izhodna napetost (rezultat)	0,45 V 2,45 V
Odzivni čas	39 ms

Pini: GND – Ozemljitev, Vcc – Napajanje, V0 – izhodna analogna napetost

Na sliki 6 je blok shema senzorja. Senzor priklopimo na Arduino s 3-polnim konektorjem.



Slika 6 – Blok shema senzorja Sharp GP2D12

3.2.2 Izračun razdalje na podlagi izhodne napetosti senzorja

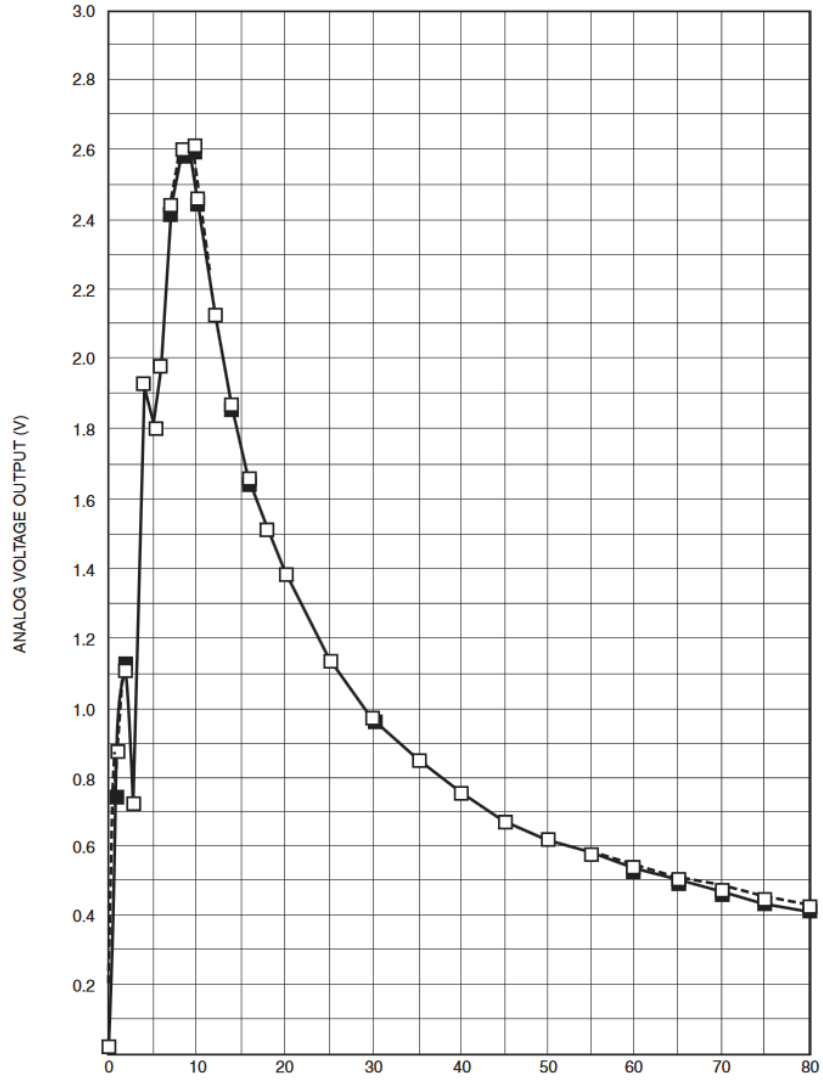
Arduino omogoča meritev analogne vrednosti na enem od analognih vhodov (z A/D-pretvorbo). Mikrokrmilnik ima 10-bitni A/D-pretvornik (angl. *ADC – Analog to Digital Converter*) z napetostnim območjem od 0 do 5 V. 10-bitni A/D-pretvornik lahko razbere 2^10 = 1024 različnih stanj.

Okolje Arduino poskrbi, da je meritev transparentna – izhodno napetost senzorja na analognem vhodu preberemo s klicem funkcije analogRead(). Funkcija vrača vrednosti od 0 do 1023 (razpon 1024). To zalogo vrednosti pa je potrebno pretvoriti v napetost. Ker je porazdelitev linearna (0 V = 0 in 5 V = 1023), lahko merjeno napetost izračunamo po formuli:

$$Merjena\ napetost\ (Vin) = \frac{Izmerjena\ diskretna\ vrednost * 5\,V}{1023}$$

Ko dobimo merjeno napetost (V_{in}), je potreben še en korak, da napetost pretvorimo v razdaljo. Na sliki 7 je graf, ki prikazuje odvisnost izhodne napetosti senzorja od razdalje.

Razdaljo bo preračunal mikrokrmilnik na podlagi izmerjene napetosti. To bomo storili tako, da bomo z grafa odčitali 14 različnih točk, vmesne točke in s tem vrednosti bodo interpolirane.



Slika 7 – Odvisnost izhodne napetosti senzorja od razdalje

3.3 Priprava programa za odčitavanje

3.3.1 Odčitavanje analogne vrednosti

Za boljšo natančnost meritve je priporočljivo meritev nekajkrat ponoviti. Funkcija za merjenje analogne vrednosti na izbranem vhodu ponovi meritev desetkrat.

```
int readAnalog(int pin)
{
  int aRead=0;
  int count=10;
  for(int i=0;i<count;i++)
  {
    aRead=aRead+analogRead(pin);
```

```
  }
  aRead=aRead/count;
  return aRead;
}
```

3.3.2 Pretvorba v razdaljo

Prebrano analogno vrednost najprej spremenimo v napetost s pripravljeno funkcijo AnalogToVoltage().

```
float AnalogToVoltage(int analog)
{
  float voltage=0;
  voltage=(5*analog/float(1023));
  return voltage;
}
```

Vrednost, ki jo zgornja funkcija vrne, pošljemo v funkcijo VoltageToCm(). Ta funkcija je prilagojena karakteristiki senzorja in vrne razdaljo v cm. Funkcija uporablja dva globalno deklarirana seznama, in sicer arrVoltage[] in addDistance[], ki hranita podatke točk.

```
float arrVoltage[14] = {2.6, 2.5, 2.35, 2,    1.5, 1.3, 0.95,  0.83,   0.74, 0.6,
0.515,   0.46,  0.45,  0.43};
float arrDistance[14] ={8,   9,   10,   12.5, 17,  20,  30,    35,    40,   50,
60,      70,      75,    80 };

float VoltageToCm(float voltage)
{
   if (voltage>arrVoltage[0]) return 0;
   if (voltage<arrVoltage[13]) return 0;

   for (int i=0;i<=13;i++)
   {
     if (voltage==arrVoltage[i]) return arrDistance[i];
     if (voltage==arrVoltage[i+1]) return arrDistance[i+1];

     if (voltage<arrVoltage[i] && voltage>arrVoltage[i+1])
     {
       float diffV=arrVoltage[i]-arrVoltage[i+1];
       float voltagePos=voltage-arrVoltage[i+1];
       float proc=1 - voltagePos/diffV;

       float diffCm=arrDistance[i+1]-arrDistance[i];
       float sum = diffCm*proc;
       float result = arrDistance[i] + sum;
       return result;
     }
   }
}
```

Poglavje 4 Pošiljanje podatkov v center

Ko strojna oprema oziroma v nadaljevanju tudi merilnik nivoja uspešno razbere podatke senzorja, je čas, da jih posreduje strežniku. Poglavje obravnava različne možnosti pošiljanja podatkov in implementacijo izbranega načina.

4.1 Možnosti za pošiljanje podatkov

»Center« imenujemo domači oziroma hišni strežnik pri uporabniku, kjer želimo podatke hraniti (strežnik MySql) in do njih kasneje tudi dostopati (spletni strežnik). Z uporabniškega stališča ne bi bilo tako praktično (vsekakor pa izvedljivo), če bi strojna oprema za merjenje pošiljala podatke neposredno do končnega uporabnika (na primer obvestila SMS z modulom GSM).

Širše možnosti omogoča implementacija strežniškega sistema in podatkovne baze, kjer se podatki hranijo ter so tako na voljo in vpogled tudi več različnim odjemalcem. Kot dodatno prednost tako gradimo zgodovino oziroma pridobivamo trend nihanja vode, kar lahko privlačno prikažemo v grafični obliki in ponudimo vpogled v dogajanje v poljubnem časovnem intervalu.

Razdalja od vodnega zajetja in s tem strojne opreme do centra (strežnika) je cca 1 km, pri čemer gre za vidno razdaljo. Če bi bila razdalja relativno kratka (nekaj 10 m), bi lahko enoto povezali kar na domači Wi-Fi, tako pa so možnosti za vzpostavitev komunikacijskega kanala predstavljene v nadaljevanju.

4.1.1 Povezovanje z modulom ZigBee

Privlačno možnost povezave ponuja tehnologija ZigBee, ki temelji na standardu IEEE 802.15.4. Tehnologija je razvita posebno z namenom povezovanja domačih in industrijskih naprav na krajših razdaljah. Ker gre v osnovi za industrijski protokol, lahko bi rekli industrijski Wi-Fi, sta bili eden glavnih ciljev tudi robustnost in varnost, zato 128-bitna enkripcija AES.

Namen tehnologije ZigBee je, da je enostavnejša in cenejša od drugih možnosti mreženja, kot sta na primer Bluetooth in Wi-Fi. Tako lahko tehnologijo uporabljajo manjše domače ali industrijske naprave na krajšem dosegu, kjer hitrost ni tako ključna, ključna pa je nizka poraba energije in možnost dodajanja novih točk.

Tovrstno mreženje je prav zaradi svoje narave priljubljeno pri nadzorovanju okolja (temperatura, vlaga, osvetljenost, onesnaženost in tako dalje) oziroma na splošno pri telemetriji.

Slika 8 – Modul Xbee Pro z žično anteno

Čeprav je bil sistem ZigBee prvotno mišljen za sorazmerno kratke razdalje, lahko zdaj dobimo module ZigBee z zunanjo anteno, ki brez težav komunicirajo na razdalji več kilometrov, če vmes ni posebnih ovir.

Izvedb ZigBee-ja je razumljivo več in glede na potrebe lahko izberemo ustrezen modul. Podjetje Digi ima na primer izvedenke, kot so:

- XBee – modul primeren za manjše razdalje do 100 m, oddajna moč 1 mW, poraba 45 mA. Hitrost prenosa do 250 kb/s. Cena 22 EUR.

- XBee Pro – modul z močnejšim signalom, ki lahko dosega razdalje 1,5 km ali več (odvisno od antene). Modul ima lahko anteno v več izvedbah. Pri konektorju RPSMA[4] (ločena antena) je poraba nekoliko večja. Oddajna moč je 63 mW, poraba 40–50 mA, tok ob oddajanju do 250 mA (340 mA za RPSMA). Hitrost prenosa je do 250 kb/s. Cena 33 EUR in 39 EUR za RPSMA.

- XBee Pro 900 RPSMA – posebna izvedba XBee s konektorjem RPSMA, ki deluje na frekvenci 900 MHz in ima domet do 10 km. Hitrost prenosa je 156 kb/s. Cena 48 EUR.

Vse zgornje izvedenke delujejo v industrijskem temperaturnem območju (–40 °C do +85 °C).

4.1.2 Povezovanje z modulom GSM/GPRS

GSM/GPRS-izmenjava podatkov je dobro poznan način za komunikacijo oziroma vzpostavitev povezave do interneta, s tem pa tudi do vseh spletišč. Platforma Arduino ima kar nekaj že

[4] SMA (angl. *SubMiniature version A*) je pomanjšana različica RF-konektorja za koaksialni kabel. RPSMA je oznaka za obratno polariteto (angl. *reverse polarity*).

vnaprej pripravljenih polno operativnih modulov GSM/GPRS, ki se na ploščo Arduino samo nataknejo in že omogočajo komunikacijo (na primer posredovanje ali prejemanje sporočil SMS ali podatkovno povezavo preko GPRS).

Slika 9 – Razširitveni modul Arduino GSM/GPRS

Na sliki 9 je razširitveni modul Arduino. S takim modulom lahko Arduino na spletu poljubno nastopa kot strežnik ali odjemalec. Pogoj za povezavo je kartica SIM (angl. *Subscriber Identity Module*) ter seveda ustrezno predplačniško ali naročniško razmerje. Modul upravljamo z ukazi AT. Ukazi AT (angl. *ATention*) se uporabljajo pri modemih in podobnih napravah. Vsak ukaz se začne z nizom AT, ki nakazuje ukazno vrstico.

Modul GSM/GRPS potrebuje dovolj močno in stabilno napajanje. Modul lahko povežemo neposredno na 5 V-napajanje Arduina, treba pa je upoštevati, da ob vzpostavitvi povezave čip SIM900 (na modulu) zahteva tudi do 2 A toka (največja konica pa lahko doseže 3 A). Poraba moči je 1 ali 2 W (odvisno od frekvenčnega območja GSM), temperaturno območje od 30 °C do +80 °C. V stanju pripravljenosti je poraba 22 mA, pri podatkovnem klicu GPRS pa od 300 mA do 400 mA.

4.2 Tehnologija ZigBee

Za našo napravo smo izbrali način komunikacije s tehnologijo ZigBee, pri čemer so ključne prednosti:

- majhna poraba energije (manjša kot pri GSM/GPRS-modulu),
- nizka cena prenosa podatkov (ni stroškov prenosa razen porabe energije),

- neodvisnost od operaterja.

V Evropi deluje ZigBee na nelicenčnem frekvenčnem območju ISM (angl. *Industrial, Scientific and Medical*) z 2,4 Ghz. Podpira različne mrežne topologije, kot so:

- **točka v točko (angl. *Point-to-Point*)**

  Uporabno za povezavo ene naprave do druge. Tukaj velja, da sta obe napravi enakovredni.

- **točka v več točk (angl. *Point-to-Multipoint*)**

  Uporabno za povezavo več točk (naprav) do centra (strežnika). Pri tem velja, da so vse naprave povezane do osrednjega vozlišča (centra), ki ga tu imenujemo koordinator. Koordinator tudi skrbi za komunikacijo dveh točk, torej posluša podatke ali posreduje podatke do točk. Slabost tovrstne topologije je morebiten izpad koordinatorja, ki je ključen pri komunikaciji poljubnih parov točk.

- **zankasto omrežje (angl. *mesh network*)**

  Zankasto omrežje odpravlja zgornjo pomanjkljivost enega koordinatorja. Tukaj lahko vse naprave komunicirajo tudi medsebojno. Vsaka novo dodana točka se sama vključi v omrežje (število točk gre lahko do 65.000). Tako omrežje je zelo prilagodljivo in lahko pokriva večja območja.

4.3 Modul XBee Pro

Za komunikacijo smo izbrali izvedbo modula XBee Pro podjetja Digi. Tehnični podatki so v tabeli 4.

Tabela 4 – Tehnične karakteristike modula XBee Pro

Oznaka	XBee Pro
Domet (sobno)	90 m
Domet na prostem	1,5 km
Oddajna moč	63 mW
Hitrost prenosa	250 kb/s
Hitrost serijske povezave	1200–250 kb/s
Napajalna napetost	2,8–3,4 V

Tok pri pošiljanju	250 mA
Tok v stanju pripravljenosti	55 mA
Frekvenca	2.4 GHz
Dimenzija	2,4 cm x 3,3 cm

4.4 Komunikacija Arduino-XBee

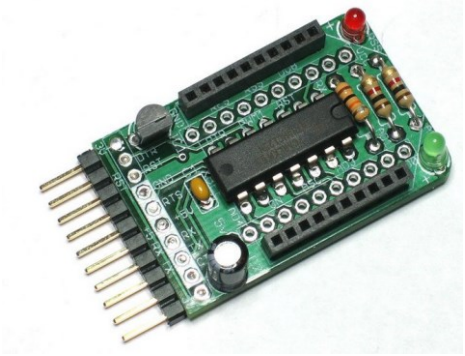
Čeprav lahko modul XBee napajamo z reguliranim izhodom Arduino za 3,3 V, bomo kot vmesnik uporabili dodaten adapter, predvsem iz dveh razlogov:

- ker želimo, da je mogoče XBee napajati tudi iz baterije oziroma za primer, ko plošče Arduino ne bi imeli in s tem tudi ne 3,3 V-regulatorja napetosti (neodvisen mikrokrmilnik ATmega323P);
- enostavna možnost povezave tudi na strežniku. Drugi modul je potrebno povezati z računalnikom, zaželeno je, da lahko to uredimo z vmesnikom USB.

Kot rešitev je izbran enostaven adapter, katerega načrt dobimo na spletu [7] in ga lahko enostavno sestavimo. Pri nakupu se cena giblje pod 10 EUR. Adapter ponuja:

- 3,3 V-regulator napetosti, ki zagotavlja tok do 250 mA, tako da lahko XBee priklopimo tudi na 5 V,
- dve LED diodi, ki nakazujeta aktivnost modula XBee (zelena – stanje pripravljenosti, rdeča – branje podatkov),
- kompatibilnost s kablom FTDI[5] in s tem enostavno povezavo na vmesnik USB.

[5] Kabel FTDI – pretvornik za povezavo RS-232 na USB. Gonilnik za uporabo FTDI je mogoče pridobiti na domači strani http://www.ftdichip.com/.



Slika 10 – Adapter XBee

Predvsem možnost povezave na vhod USB je izredno uporabna, saj lahko XBee enostavno priklopimo na računalnik in z njim komuniciramo. To potrebujemo za strežniški del in ne nazadnje taka povezava olajša testiranje in konfiguracijo modula XBee.

4.5 Konfiguracija XBee

Na voljo imamo dva modula XBee Pro, ki ju moramo povezati. Ustrezno konfiguracijo modula XBee lahko opravimo z računalnikom in kablom FTDI, ki je povezan na adapter XBee.

Najbolj enostavna vrsta povezave je povezava točka v točko (angl. *Point2Point*), ki jo bomo uporabili tudi v tem primeru. Rezultat bo transparenten most med strojno opremo in strežnikom (pin Tx prvega modula se poveže na Rx drugega in obratno). Modul XBee ima kar nekaj možnosti za konfiguracijo, kot na primer omrežni naslov, način delovanja, režim varčevanja z energijo, hitrost serijske povezave, izhodna moč, ključ za enkripcijo in tako dalje. Vse možnosti so razvidne v dokumentaciji [8].

Za povezavo modulov je ključen enak naslov omrežja (angl. *network ID*). Omrežni ID, poznan tudi kot PAN ID (angl. *Personal Area Network ID*), lahko preberemo in nastavimo z ukazi AT. Za konfiguracijo potrebujemo serijski terminal, na primer Putty, pri čemer za vzpostavitev ukaznega načina vpišemo ukaz +++ in modul odgovori z OK. Za oba modula nastavimo PAN ID, kot je v naslednjem koraku.

ukazi – modro, odgovori – rdeče

```
+++OK
ATID (pridobi trenutni PAN)
3332 (privzeta vrednost)
ATID 2525 (nastavimo nov PAN ID)
OK
ATID (preverimo)
2525
```

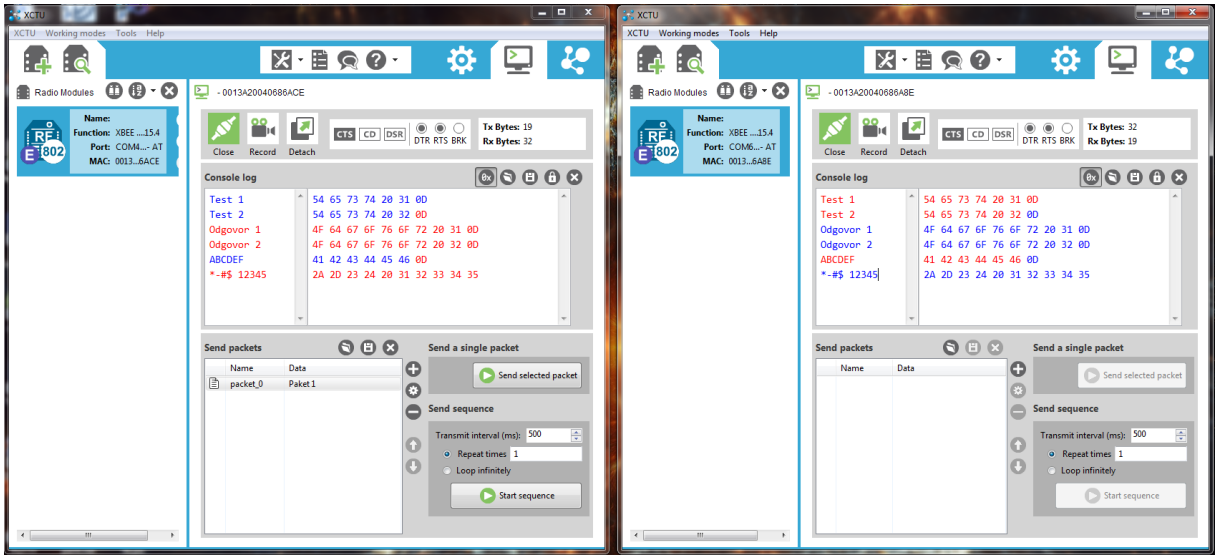
```
ATWR (zapišemo na flash)
OK
```

4.5.1 Uporaba orodja XCTU

Lažji in boljši način za konfiguracijo ter kasnejše testiranje je uporaba namenskega in brezplačnega orodja XCTU podjetja Digi. XCTU je aplikacija v okolju Windows, namenjena posebej testiranju in razhroščevanju, omogoča pa tudi enostavno nadgradnjo strojne programske opreme (angl. *firmware*).

Program prepozna napravo in prebere vse atribute, ki so na voljo. Vrednosti lahko tudi ponastavimo na tovarniško nastavitev, si naredimo različne profile konfiguracij in to shranimo nazaj na modul. Program ima tudi konzolno okno za testiranje serijske povezave.

Primer testa povezljivosti med eno in drugo enoto je na sliki 11. Pri tem sta oba modula XBee preko adapterja in kabla FTDI povezana na isti računalnik. Program XCTU in konzolno okno sta odprta za vsako napravo posebej.



Slika 11 – Test komunikacije z orodjem XCTU

Na sliki 11 vidimo moder in rdeč tekst. Modro je poslano – rdeče je prejeto.

4.6 Pošiljanje podatkov

Arduino oziroma mikrokrmilnik povežemo preko adapterja XBee na modul XBee. Za serijsko povezavo uporabimo pina Rx in Tx. Arduino že ima krmilnik UART za serijsko povezavo na pinu 0 in 1 (Rx, Tx). Ker se ta povezava uporablja za programiranje in za pisanje na konzolo,

bomo za XBee uporabili pina 2 in 3. To lahko dosežemo z uporabo knjižnice, ki emulira UART, na primer SotwareSerial, in je priložena okolju Arduino.

Podatek o meritvi pošljemo kot tekstovni niz v obliki:

```
#::{TIP}::{ID}::{Meritev_ORG}::{Meritev_V}::{Meritev_D}::{VM}@
```

Pomen parametrov:

- TIP – tip podatkovnega paketa (M – meritev),
- ID – zaporedna številka meritve (ki se v centru uporablja kot indikator, če je kaka meritev izgubljena),
- Meritev_ORG – originalna digitalna vrednost, prebrana iz senzorja,
- Meritev_V – meritev pretvorjena v napetost,
- Meritev_D – meritev pretvorjena v razdaljo,
- VM – vrsta meritve, pri čemer velja: 0 – redna ciklična meritev, 1 – meritev posredovana, ker je doseženo odstopanje, 2 – meritev zahtevana.

Primer podatkovnega paketa za meritev je:

```
#::M::1::480::2.35::10::0@
```

4.7 Prejemanje zahtevkov

Poleg pošiljanja podatkov o meritvah bomo strojno opremo pripravili tudi na sprejem podatkov. Zahtevke, ki jih lahko strežnik pošlje strojni opremi, razdelimo na:

- zahtevek za zadnjo meritev,
- zahtevek za trenutno meritev,
- ponastavitev parametrov delovanja.

Pri serijski povezavi na XBee tako implementiramo pošiljanje in sprejemanje. Če na vhodu zaznamo niz znakov, te poskušamo sestaviti v polni ukaz, upoštevajoč kontrolna znaka za

začetek in konec. Ko zaznamo kontrolni znak za konec ukaza, je ukaz pripravljen za interpretacijo. Tovrstni režim postopnega zlaganja znak za znakom je potreben, saj ni zagotovljeno, da celoten ukaz prispe v enem obhodu zanke.

```
void readZigBee()
{
      while (ZigBee.available())
      {
        char msg = (char)ZigBee.read();

        if (msg=='@')
        {
          inMessage = "";
          bNewInMessage = false;
          continue;
        }

        if (msg=='#')
        {
          bNewInMessage = true;
          continue;
        }

        inMessage +=msg;
      }
}
```

4.7.1 Zahtevek za zadnjo meritev

S tem zahtevkom pošljemo strojni opremi ukaz, da ponovno pošlje zadnjo meritev. Gre za neke vrste »echo« zahtevek, s katerim tudi ugotovimo, ali je strojna oprema operativna. Na strani strežnika bo zahtevek uporaben tudi za testiranje sprejema podatkov.

Ukaz:

```
#::REQ::M@
```

4.7.2 Zahtevek za trenutno meritev

Zahtevek sproži ponovno merjenje nivoja vode in posreduje meritev na strežnik v klasični obliki. S tem zahtevkom lahko odjemalec v vsakem trenutku pridobi svežo meritev.

```
#::REQ::MC@
```

4.7.3 Ponastavitev parametrov delovanja

Strojna oprema ima nekaj nastavitvenih parametrov, ki so zapisani v programu in imajo privzete vrednosti. To so:

- **interval merjenja** – interval izvajanja meritev, privzeto je 5 min,
- **interval pošiljanja –** interval pošiljanja meritev na strežnik, privzeto je 60 min,
- **največja sprememba** – največja dovoljena sprememba med zadnjo in trenutno meritvijo. Če je razlika večja, se meritev pošlje v center ne glede na interval pošiljanja.

Vsi zgornji parametri so zapisani na Arduinu oziroma na čipu ATmega328P ob programiranju. Ker bo strojna oprema na terenu in želimo dopustiti tudi možnost kasnejšega ponastavljanja parametrov, funkcija za kalibracijo preko mostu XBee omogoča ponastavitev privzetih vrednosti brez ponovnega programiranja čipa.

Ukaz za ponastavitev parametrov

```
#::REQ::{int_m}::{int_p}::{max_diff}@
```

Poglavje 5 Merilnik nivoja

V tej točki poznamo vse nujno potrebne elemente strojne opreme oziroma merilnika nivoja in imamo tudi že nekaj izhodiščnih funkcij. V nadaljevanju bosta predstavljena vezalna shema in diagram poteka programa. Hkrati bomo strojno opremo nekoliko razširili, kar bo omogočilo lažje testiranje in tudi avtonomnejše delovanje.

5.1 Testni način delovanja

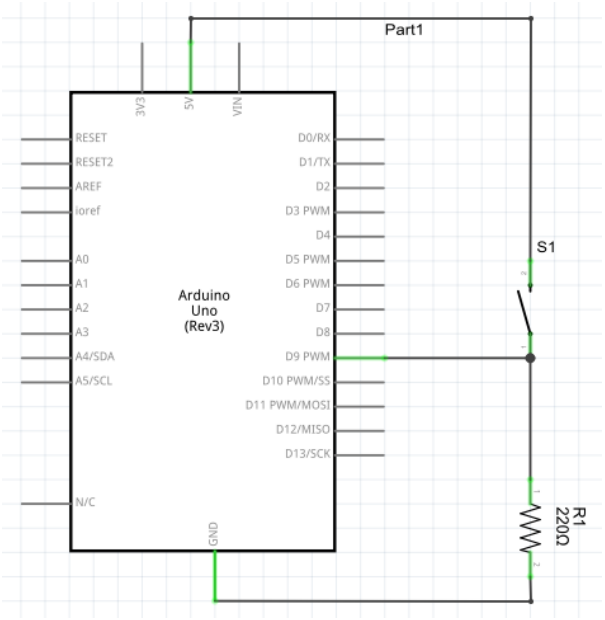
Vpeljali bomo več načinov delovanja strojne opreme:

- **normalni (delovni način)** – v tem načinu se privzeto meritev opravlja na 5 min in meritev pošilja na strežnik na 60 min. Če je odstopanje večje od dovoljenega, se meritev pošlje tudi prej. S tem zagotovimo, da meritve na strežnik ne prihajajo prepogosto ali po nepotrebnem, hkrati pa pride meritev pri večji spremembi;

- **testni način** – v testnem načinu strojna oprema opravlja meritev razdalje na 1 s in rezultat prikazuje na privzeti serijski povezavi (UART) ter istočasno na LCD-prikazovalniku, če je priklopljen;

- **testni način s pošiljanjem meritev** – v testnem načinu strojna oprema opravlja meritev razdalje na 1 s in podatke prikazuje, hkrati pa na 1 min podatke o meritvah tudi pošilja na strežnik.

Preklop režima delovanja bo zagotovljen z branjem zunanjih vhodnih digitalnih vhodov, na katere bo povezano DIP-stikalo. DIP-stikalo ima dve stanji – sklenjen ali razklenjen kontakt. Stikalo z uporom vežemo tako, da je privzeto (razklenjen kontakt) na Arduinu izbran digitalni vhod na potencialu 0 V (ozemljitev). Pri preklopu stikala na Arduino dobimo 5 V oziroma logično 1, kar zaznamo v programu.



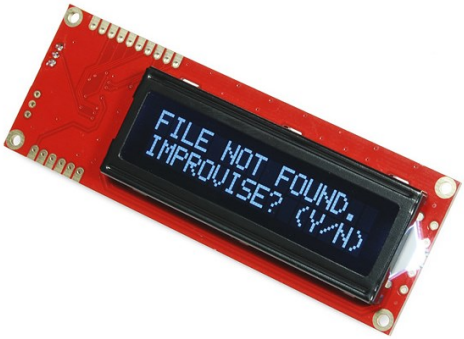
Slika 12 – Vezalni načrt DIP-stikala

5.2 Prikaz podatkov na LCD-zaslonu

Želimo si, da strojno opremo razširimo še z dodatnim LCD-prikazovalnikom. Ta bo v uporabi v testnem načinu in takrat, ko razvojno okolje Arduino ni na voljo (dislocirano testiranje ali demonstracija merjenja).

Izbira LCD-zaslona:

- izberemo standardni LCD-zaslon velikosti 2 x 16 polj,
- izberemo LCD-zaslon, ki ima podporo za serijsko povezavo. Tako za komunikacijo potrebujemo le en signalni vodnik (LCD-vhod Rx povežemo na Arduino izhod Tx).



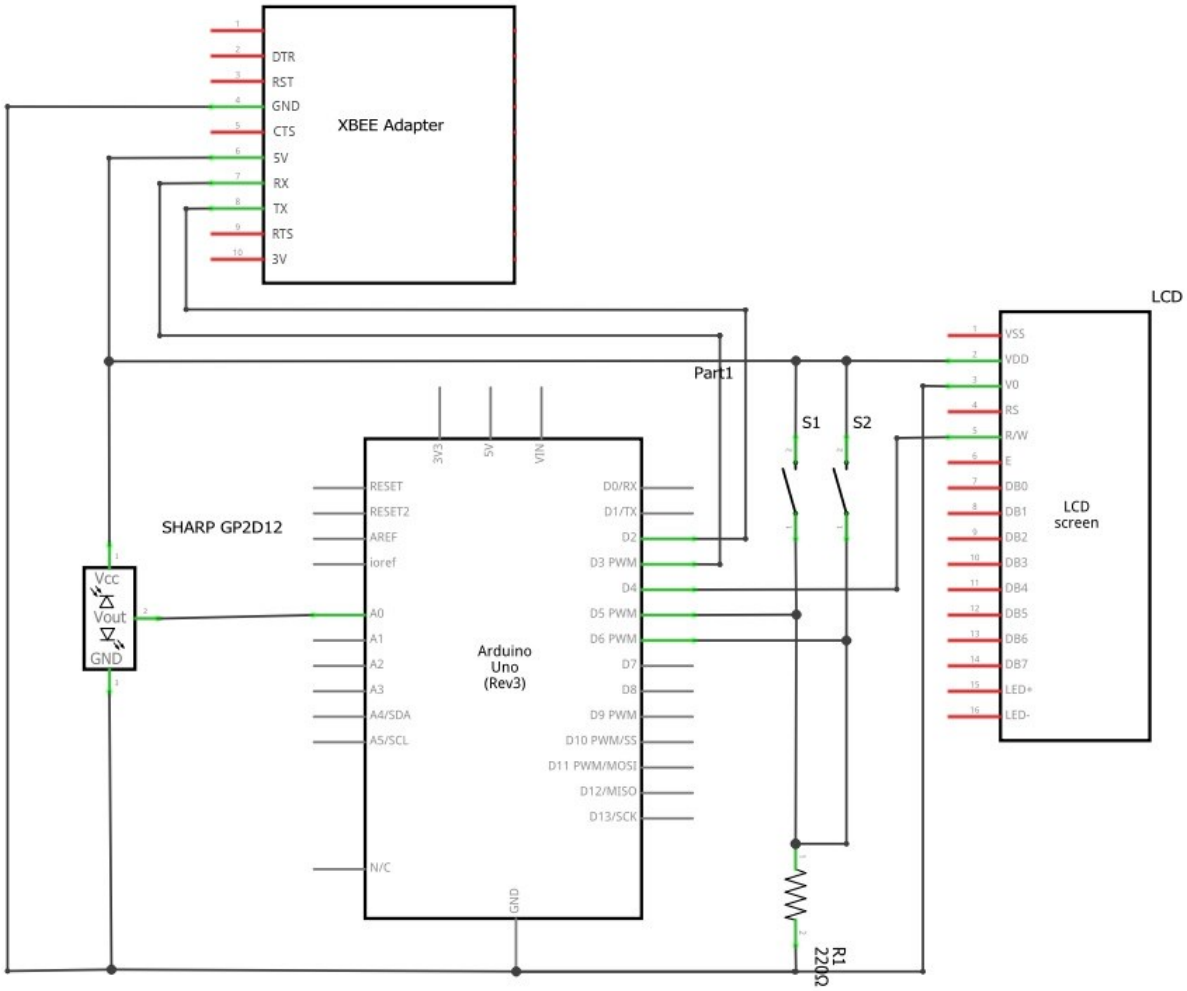
Slika 13 – LCD-zaslon

Izbrali smo LCD-zaslon, ki ima za osnovo čip PIC 16F88. Vgrajeni PIC skrbi za prikaz znakov, ki jih pošljemo po serijski TTL-povezavi. Zaslon ima tudi posebne kontrolne znake, s katerimi na primer nastavljamo osvetlitev, izbiramo vrstico in zaslon počistimo.

LCD potrebuje napajalno napetost 5 V, podpira standardne hitrosti serijske povezave med 2400 in 38.400 b/s in ima tudi možnost varčevanja z energijo (manjša osvetlitev).

5.3 Vezalna shema strojne opreme merilnika

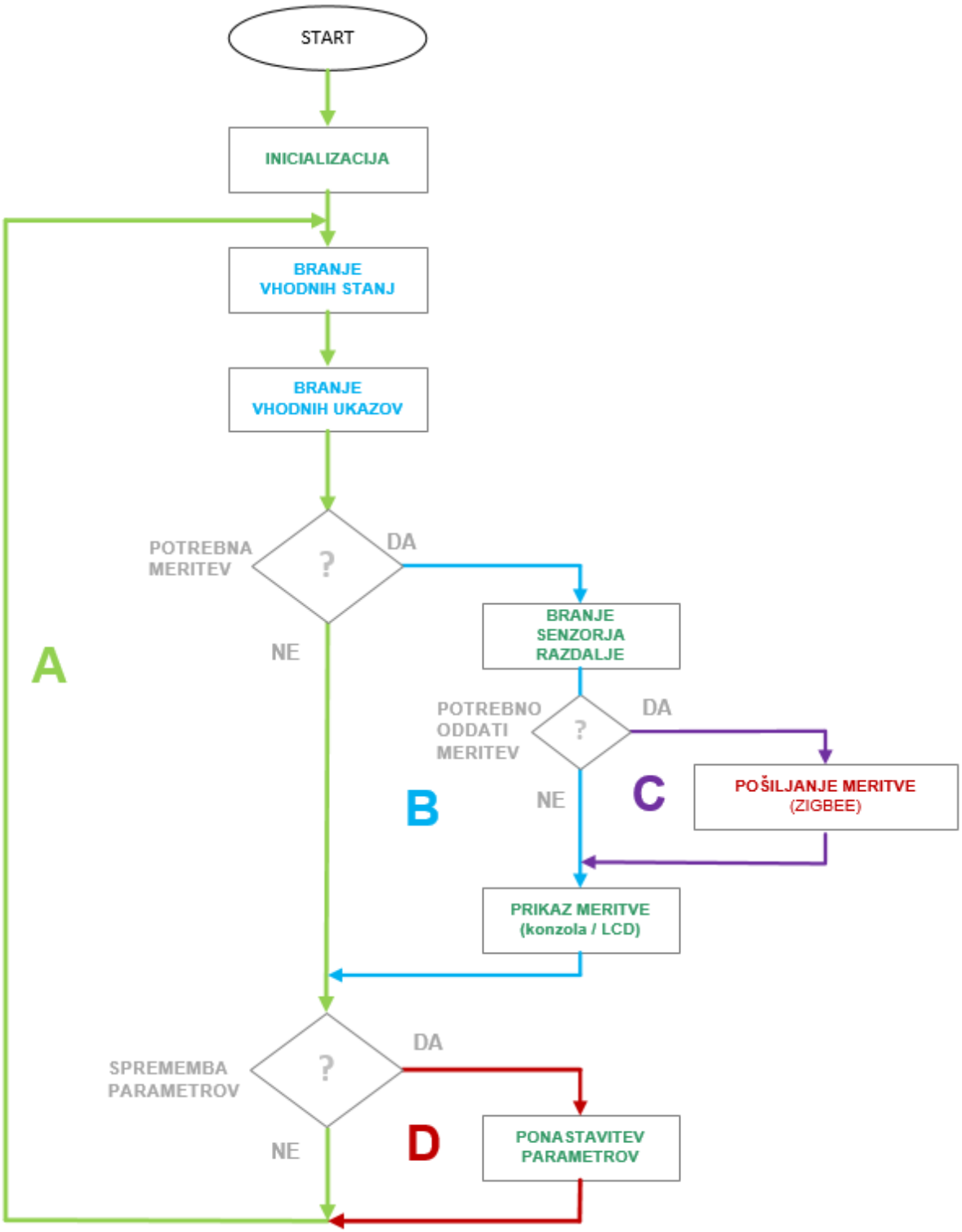
Polna vezalna shema strojne opreme merilnika je na sliki 14.



Slika 14 – Vezalna shema strojne opreme merilnika

Strojna oprema merilnika vključuje Arduino UNO kot osnovo, senzor razdalje Sharp GP2D12, adapter XBee, povezan na modul XBee, LCD-zaslon in dve vhodni stikali za preklapljanje režima delovanja naprave.

5.4 Diagram poteka programa na mikrokrmilniku



Slika 15 – Diagram poteka programa na mikrokrmilniku

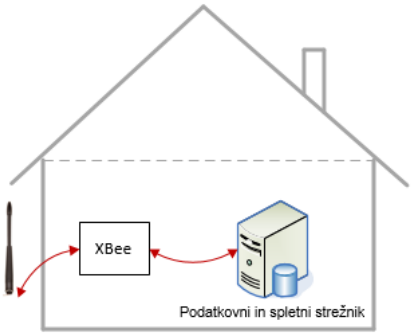
Na sliki 15 je prikazan diagram poteka programa na mikrokrmilniku. Pomen oznak na diagramu poteka je:

- zeleno A – glavna zanka programa, ki se izvaja neprekinjeno;
- modro B – metoda za odčitavanje vrednosti senzorja in pretvorbo. Meritev se privzeto izvaja na 5 min;
- vijolično C – metoda za pošiljanje podatkov v center;
- rdeče D – metoda za ponastavitev parametrov, ko je na vhodu zaznan zahtevek za ponastavitev strojne opreme.

Poglavje 6 Strežnik pri uporabniku – center

6.1 Shematski prikaz povezave

Na sliki 16 je shematski prikaz povezave na strani centra – strežnika z oddaljenim merilnikom.



Slika 16 – Povezava strežnika v centru

Računalnik, ki se uporablja kot strežnik, je preko vmesnika USB povezan na adapter XBee, ta pa preko priključka RPSMA do antene, ki jo postavimo na zunanjo stran hiše zaradi boljšega sprejema.

6.2 Podatkovna baza

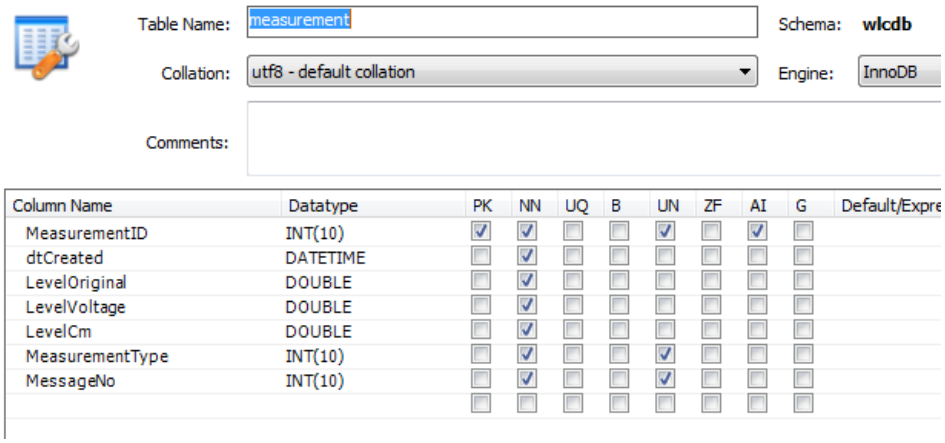
Za zbiranje podatkov smo uporabili podatkovno bazo MySql. Gre za dobro poznano in odprtokodno implementacijo relacijske podatkovne baze, ki za delo s podatki uporablja jezik SQL. MySql je pisan v jeziku C in C++ in na voljo za različne platforme, kot so Windows, Linux, OS X, AIX, FreeBSD, SunOS in tako dalje. V našem primeru bomo podatkovno zbirko uporabljali v sistemu Windows.

6.2.1 Podatkovna tabela za meritve

Podatkovno strukturo za meritve enostavno ustvarimo v okolju MySql Workbench, ki je priloženo namestitvi strežnika MySql oziroma ga lahko namestimo tudi kot neodvisno aplikacijo.

Naša podatkovna tabela za rezultate meritev (slika 17) bo vsebovala:

- vse podatke o meritvi, ki jih pošlje merilnik nivoja,
- čas sprejema podatkov, ki se določi na strežniku in nakazuje tudi čas veljavnosti.



Slika 17 – Podatkovna tabela za rezultate meritev

6.3 Programska oprema

Programska oprema na strani strežnika bo pisana v okolju .NET. Microsoft ponuja več možnosti in vlog za aplikacije, pisane v okolju .NET. Uporabljali bomo programski jezik C#, ki je med razvijalci dobro poznan in tudi cenjen. Verzija C# 1.0 izhaja iz leta 2002, zdaj pa imamo na voljo že verzijo C# 6.0 (.NET Framework 4.6).

V našem primeru bomo za strežniško aplikacijo uporabili .NET Framework 4.5 in programsko okolje Visual Studio 2012.

Programska oprema za zajem podatkov bo pisana kot servis. Servisi v okolju Windows omogočajo avtonomni način delovanja in se zaženejo avtomatično ter ne potrebujejo uporabniške interakcije. Gre za storitve operacijskega sistema, za katere je značilno, da opravljajo stvari v ozadju (uporabniku ni treba biti prijavljen).

6.4 Servis za zajem rezultatov meritev

Servis za zajem rezultatov poimenujemo kot WLCService (angl. *WaterLevelCollectorService*). Naloge, za katere je servis zadolžen, so:

- avtonomno delovanje ves čas, ko je vključen računalnik,

- spremljanje vhodnih podatkov na COM-portu (sprejem merilnih rezultatov),
- shranjevanje podatkov v podatkovno bazo MySql,
- pošiljanje podatkov na COM-port (ponastavitev parametrov ali zahtevek za meritev).

6.4.1 Priprava servisa

Servis Windows v okolju .NET implementiramo tako, da naredimo razred, ki je izpeljan iz razreda System.ServiceProces.ServiceBase. Nujni metodi za implementacijo sta dve, in sicer OnStart() in OnStop(), kot je razvidno na sliki 18. Namen metod je nakazan že v imenu – te funkcije pokliče operacijski sistem ob zagonu oziroma ustavitvi servisa.

```
namespace Water_Level_Colector_Service
{
    partial class WaterLevelCollectorService : ServiceBase
    {
        ServiceMain mainService = new ServiceMain();

        public WaterLevelCollectorService()
        {
            InitializeComponent();
        }

        protected override void OnStart(string[] args)
        {
            mainService.Start();
        }

        protected override void OnStop()
        {
            mainService.Stop();
        }
    }
}
```

Slika 18 – WLCService

Podrobne implementacije namenoma ne dajemo v razred servisa, temveč jo bomo zapisali v poseben razred, imenovan ServiceMain. Tukaj bo glavni del programa, ki ga bo naš servis izvajal in bo prav tako imel metodi Start() in Stop().

Razlog za tako zgradbo je v lažjem testiranju in razhroščevanju servisa. Ker servisi Windows nimajo uporabniškega vmesnika (angl. *GUI*), direktna interakcija ni mogoča. Servis lahko na primer upravljamo le preko zunanjih konfiguracijskih datotek ali rezultate spremljamo na izhodnih enotah, kot so datoteke ali baza.

WLCService je tako raje razvit kot hibrid med servisom in navadno aplikacijo. Tovrsten način ne zahteva veliko prilagoditev, omogoča pa, da servis med razvojem poganjamo kot klasično aplikacijo (angl. *Windows Forms Application*) in morebitne rezultate spremljamo direktno na konzoli. Windows Forms ponuja enostavno možnost proženja metod preko grafičnega vmesnika, konzola pa je univerzalni kanal za izpisovanje rezultatov.

6.4.2 Razred COMPort Manager

Po pregledu obsega servisa je jasno, da je njegov ključni del povezava na COM-port in s tem branje in pisanje podatkov. Navidezni COM-port (angl. *virtual COM port*) ustvari gonilnik za kabel FTDI, preko katerega je na računalnik vezan modul XBee. Povezava na ta COM-port preko modulov XBee tako pomeni neposredno povezavo do strojne opreme oziroma merilnika.

```
class ComManager
{
    public delegate void MessageReceived(object manager, MessagePortReceivedEventArgs args);
    public event MessageReceived MessageReceivedEvent;

    Properties

    public SerialPort SerialPort;

    public ComManager() ...

    public bool Connect()
    {
        try
        {
            SerialPort = new SerialPort(pComPort, pComBaudRate);
            SerialPort.DataReceived += new SerialDataReceivedEventHandler(DataReceivedHandler);
            SerialPort.Open();
            return true;
        }
        catch (Exception ex)
        {
            Logger.Write("ComManager::Connect", ex);
            return false;
        }
    }
```

Slika 19 – Implementacija razreda ComManager

Ker je komunikacija po COM-portu vitalnega pomena, zgradimo v ta namen poseben razred ComManager. Ta razred bomo v nadaljevanju uporabljali za odpiranje in zapiranje COM-porta, pošiljanje podatkov in tudi obratno, razred bo prožil dogodke (angl. *events*), ko se na vhodu pojavijo novi podatki.

Okolje .NET že vključuje podporo za COM-porte v okviru sistemskih knjižnic System.IO.Ports.SerialPort. Na sliki 19 je del kode za implementacijo razreda ComManager.

6.4.3 Porazdeljeni sistemi

WLCService potrebuje še en zanimiv mehanizem v okviru svojega delovanja. Ker bo ta servis edini, ki bo povezan na COM-port in s tem z modulom XBee, hkrati pa velja, da servis v normalnem režimu delovanja nima grafičnega vmesnika oziroma ga uporabnik ne vidi, se pojavlja vprašanje, kako bo ta prožil nove zahtevke do strojne opreme (na primer zahtevek za meritev).

Naša želja je, da servis shranjuje vse podatke o meritvah, ki pridejo na COM-port ne glede na čas, in je tako vedno pripravljen za vhodne podatke. Po drugi strani pa želimo podatke pošiljati na strojno opremo le v izjemnih primerih, ki pa jih proži končni uporabnik preko mobilne spletne aplikacije.

Mobilna spletna aplikacija bo v produkcijskem okolju edini grafični vmesnik, ki bo na voljo končnemu uporabniku, da preverja meritve in tudi proži morebitne nove zahtevke. V nadaljevanju tako potrebujemo povezavo med mobilno aplikacijo in servisom. Uporabili bomo princip gradnje porazdeljenih aplikacij.

Porazdeljene aplikacije so aplikacije, ki lahko delujejo v različnih procesih tudi na različnih računalnikih. Ključen je mehanizem, kako te aplikacije oziroma njihove objekte povezati. Objektom, ki so na drugem računalniku, pravimo oddaljeni objekti. V našem primeru želimo iz spletne aplikacije poklicati objekt (oziroma metodo) v servisu, ki bo posredovala podatke dalje na COM-port.

6.4.4 Tehnologija .NET Remoting

Okolje .NET ponuja več možnosti za gradnjo porazdeljenih aplikacij. Uporabili bomo tehnologijo .NET Remoting [9], ki vpeljuje veliko mero abstrakcije in zmanjša kompleksnost pri izdelavi in upravljanju oddaljenih objektov. Ko odjemalec ustvari oddaljen objekt, dobi instanco objekta na strežniku (oddaljenem procesu). V nadaljevanju velja, da je klic oddaljene metode skoraj enak klicu lokalne metode.

Za izgradnjo komunikacijskega modela .NET Remoting potrebujemo:

- objekt oziroma razred, ki služi kot vmesnik za komunikacijo (angl. *interface*) in ki definira vse oddaljene metode,
- gostiteljsko aplikacijo oziroma metodo, ki posluša za klice oddaljenega objekta (v našem primeru razred znotraj servisa WLCService),

- odjemalca, ki se poveže na gostitelja in kliče oddaljene metode (v našem primeru mobilna spletna aplikacija oziroma spletni strežnik).

6.4.4.1 Vmesnik IWLCRemotingConnector

IWLCRemotingConnector je vmesnik, ki definira oddaljene metode, definicijo vmesnika vidimo na sliki 20.

```
namespace IWLCRemotingConnector
{
    public interface IWLCRemotingConnector
    {
        string RequestConfiguration();
        bool RequestMeasurement();
        bool SendConfiguration(string message);
    }
}
```

Slika 20 – IWLCRemotingConnector

6.4.4.2 Razred RemotingServer

Razred RemotingServer skrbi za vzpostavitev komunikacijskega kanala (slika 21). V našem primeru strežnik posluša na vratih 9998 za vhodne klice. V trenutku, ko se odjemalec poveže in pride klic, se ta preusmeri na razred WLCConnector, kjer je dejanska implementacija oddaljene metode.

```
class RemotingServer
{
    public RemotingServer()...

    public void StartServer()
    {
        try
        {
            TcpChannel tcpChannel = new TcpChannel(9998);
            ChannelServices.RegisterChannel(tcpChannel,false);

            Type commonInterfaceType = typeof(WLCConnector);

            RemotingConfiguration.RegisterWellKnownServiceType(commonInterfaceType,
                "IWLCRemotingConnector", WellKnownObjectMode.SingleCall);
        }
        catch (Exception ex)
        {
            Logger.Write("StartServer", ex);
        }
    }
}
```

Slika 21 – Razred RemotingServer

6.4.5 Inicializacija servisa

Ko imamo ključne razrede pripravljene, je inicializacija servisa sorazmerno preprosta (slika 22). Takoj na začetku se povežemo na COM-port z razredom ComManager in zaženemo pripravljen oddaljeni (*Remoting*) strežnik. Metodo Start() privzeto kliče servis ob zagonu ali jo kličemo ročno pri namizni aplikaciji Windows (faza testiranja).

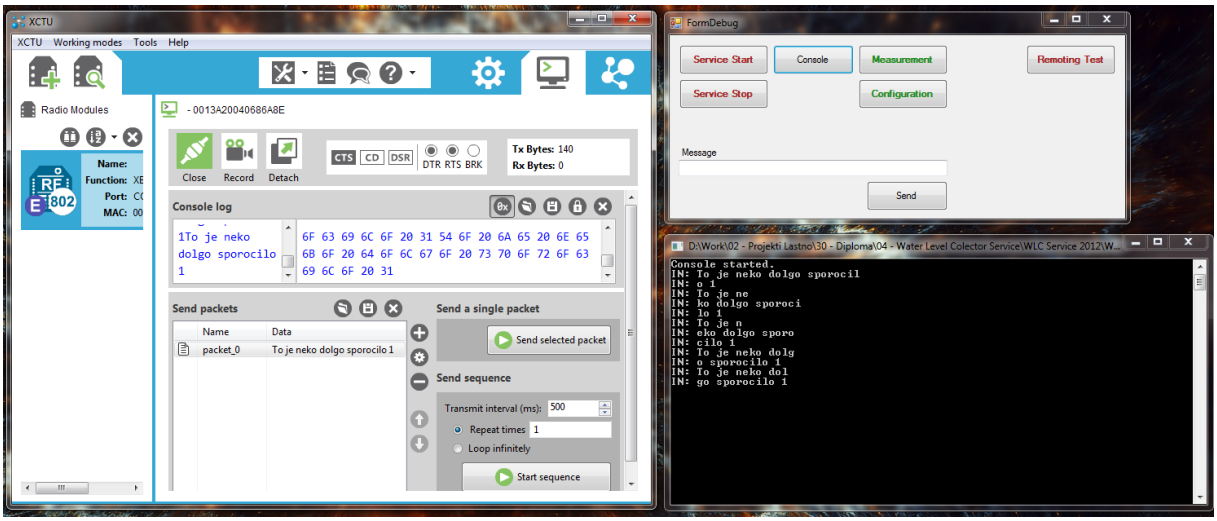
```
class ServiceMain
{
    ComManager _comManager = new ComManager();
    RemotingServer _remotingServer;

    Properties

    #region Start - Stop (service)
    public void Start()
    {
        _comManager.Connect();
        _comManager.MessageReceivedEvent += _comManager_MessageReceivedEvent;

        _remotingServer = new RemotingServer();
        _remotingServer.StartServer();
    }
```

Slika 22 – Inicializacija servisa

6.5 WLCService kot namizna aplikacija

Na sliki 23 je na levi strani orodje XCTU, ki komunicira z modulom XBee 1 (strojno opremo), in na sliki desno namizna aplikacija WLCService z odprto konzolo in oknom za upravljanje. Aplikacija je povezana na modul XBee 2 in sprejema podatke.

Na sliki 23 oziroma na konzoli vidimo, kako daljši nizi (na primer »To je dolgo sporocilo 1«) pridejo na XBee ter kasneje na COM-port v več iteracijah.



Slika 23 – WLCService sprejemanje sporočil

6.6 Sprejem podatkov

Enako kot na strani strojne opreme tudi na strani servisa za sprejemanje sporočil metodo za branje sporočil implementiramo tako, da čaka na kontrolna znaka za začetek in konec sporočila (znak # in znak @). Tako zagotovimo, da smo sprejeli celotno besedo pred njeno nadaljnjo interpretacijo.

Ko podatek razčlenimo (angl. *parsing*), razberemo numerične vrednosti posredovane meritve in jih zapišemo v podatkovno bazo. Podatki so v tem trenutku pripravljeni za pregled s strani mobilne spletne aplikacije.

6.6.1 Proženje obvestil in opozoril

V trenutku, ko smo sprejeli novo sporočilo in ga tudi zapisali v bazo, se lahko odločimo, ali je potrebna še kaka dodatna akcija na podlagi razbranih podatkov. Da bo uporabnik vsako novo

sporočilo videl v trenutku prejema, seveda ni pričakovati in to tudi ni potrebno. Istočasno pa ravno cilj celotne rešitve v pravočasni obveščenosti.

Obveščanje pri mejnih vrednostih enostavno implementiramo v okviru WLCServisa, kar je še ena njegovih prednosti. Za obveščanje je zelo praktična uporaba elektronske pošte, ki je hitra in brezplačna. Uporabnik lahko dobi elektronsko pošto neposredno na telefon, če je le priklopljen na internet. V okviru servisa v ta namen uporabimo razred System.Net.Mail, elektronsko sporočilo pa se pošlje, ko izmerjeni nivo vode v zbiralniku pade na 50 odstotkov oziroma manj. Podobno bi lahko WLCService pošiljal tudi SMS-obvestila ob pomoči katerega od temu namenjenih spletnih servisov (ki pa običajno niso brezplačni).

V okviru obveščanja lahko zagotovimo še en mehanizem pri morebitnem izpadu delovanja merilnika nivoja. Če WLCService v pričakovanem obdobju ne prejme podatkov o novi meritvi, pošlje uporabniku obvestilo o napaki. Tako smo obveščeni, če bi nastala napaka ali težava pri strojni opremi oziroma merilniku.

Poglavje 7 Mobilna spletna aplikacija

Mobilna spletna aplikacija (v nadaljevanju aplikacija) je zadnji korak v našem projektu. Za aplikacijo želimo, da omogoča:

- pregled stanja na eni ali več mobilnih napravah in na namiznem računalniku,
- pregled zadnje meritve,
- pregled zgodovine meritev,
- oddaja zahtevka za trenutno meritev,
- ponastavitev parametrov delovanja.

7.1 Tehnologija za razvoj spletne aplikacije

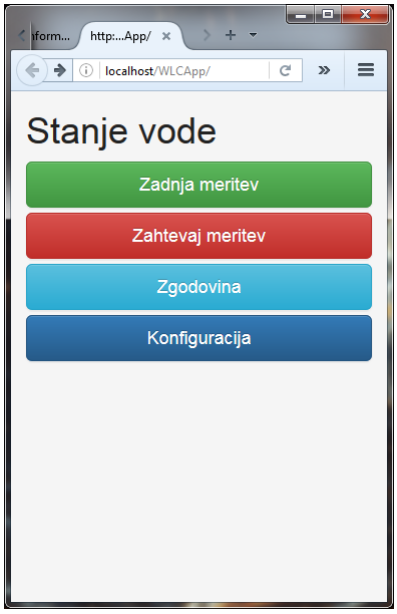
Aplikacijo razvijemo v okolju.NET, tokrat kot ASP.NET Web Application v VS 2012. Pri tem bomo uporabili tehnologije:

- HTML5/CSS3 – uporabniški vmesnik,
- Bootstrap – uporabniški vmesnik,
- jQuery – skriptni jezik na strani odjemalca,
- Ext.NET – podpora .NET za grafe na strani odjemalca.

Spletno stran gosti strežnik Windows IIS, postavljen na našem domačem strežniku, ki je povezan v omrežje. Do aplikacije ter s tem tudi pregleda in upravljanja strojne opreme lahko dostopamo s poljubno napravo od koder koli, če smo le povezani na internet.

7.2 Aplikacija – vstopna stran

Vstopna stran aplikacije je na sliki 24. Aplikacija na svoji vstopni strani ponudi uporabniku možnosti: Zadnja meritev, Zahtevaj meritev, Zgodovina in Konfiguracija.



Slika 24 – Spletna aplikacija – vstopna stran

7.3 Pregled izmerjenih rezultatov

Izmerjeni rezultati so uporabniku predstavljeni grafično. Tako zadnje stanje (stanje nivoja vode v zbiralniku) kot zgodovino meritev lahko predstavimo v grafu. Za podporo risanju grafov na strani odjemalca (spletnega brskalnika) bomo uporabili orodje Ext.NET, ki je posebej prilagojeno okolju .NET.

Ext.NET [10] je profesionalno in komercialno orodje, ki omogoča hitrejši razvoj spletnega uporabniškega vmesnika. Okolje vključuje ogromno že pripravljenih gradnikov ASP.NET AJAX (Web forms in MVC), ki temeljijo na osnovi programskega ogrodja Sencha ExtJS [11].

7.3.1 Zadnje stanje

Slika 25 – Spletna aplikacija – zadnje stanje

Na sliki 25 je prikaz zadnjega stanja nivoja vode, kot je bilo sprejeto iz merilnika nivoja. Zeleni pas zgoraj nakazuje, da je meritev »sveža« - to je stara manj kot 15 min. Če je meritev opravljena in prejeta med 15 in 60 minutami, je to območje oranžno. Če preteče od zadnje meritve več kot 60 minut, se področje obarva rdeče – kar nakazuje na napako v sistemu, saj pričakujemo redne meritve na vsakih 60 min.

7.3.2 Zgodovina meritev

Zgodovina meritev na sliki 26 prikazuje trend gibanja nivoja vode v zbiralniku, privzeto za zadnjih 30 dni. Filter pregleda lahko uporabnik tudi ponastavi in tako spremlja daljše obdobje ali podrobneje analizira krajši časovni interval.

Slika 26 – Spletna aplikacija – zgodovina meritev

7.4 Pošiljanje zahtevkov

7.4.1 Povezava na Remoting strežnik

Za posredovanje zahtevkov na strojno opremo bomo uporabili že pripravljen kanal .NET Remoting, kot je opisan v poglavju 6.4.4. Odjemalec bo v tem primeru spletni strežnik, ki dobi ukaz od končnega uporabnika preko spletne aplikacije. Gostitelj je WLCService.

7.4.2 Ponovna meritev

Za potrebe klicev oddaljenih metod bomo naredili razred, ki bo dodal dodatno abstrakcijo na strani spletnega strežnika. Razred WLCConnector bo implementiral vse metode, ki jih želimo klicati na oddaljenem objektu, in sicer tako, da bo sam vzpostavil remoting kanal in ustrezno metodo poklical. Če je bila metoda uspešno klicana, bo funkcija vračala rezultat »true«, sicer bo rezultat »false«. Primer strežniške funkcije, ki pošlje zahtevek na oddaljen objekt, je na sliki 27.

```
public class WLCConnector
{
    public bool RequestMeasurement()
    {
        try
        {
            Type requiredType = typeof(IWLCRemotingConnector.IWLCRemotingConnector);

            IWLCRemotingConnector.IWLCRemotingConnector remoteWLC =
                (IWLCRemotingConnector.IWLCRemotingConnector)Activator.GetObject
                (requiredType, "tcp://localhost:9998/IWLCRemotingConnector");

            string temp = remoteWLC.RequestConfiguration();
            return true;
        }
        catch (Exception ex)
        {
            return false;
        }
    }
}
```

Slika 27 – Razred WLCConnector za abstrakcijo klicev na strani spletnega strežnika

Kot je razvidno iz slike 27, uporabljamo protokol TCP in se povezujemo na »localhost«, ker sta oba procesa na istem računalniku. Enako bi se lahko povezali tudi na proces na drugem računalniku, ki je omrežno dosegljiv.

7.4.3 Ponastavitev parametrov

Ponastavitev parametrov je še zadnja operacija, ki jo pokrijemo v okviru mobilne spletne aplikacije. Deluje tako, da na klasičen način zajamemo vhodne podatke (*HTML form*) in jih posredujemo na strežnik. Strežnik pripravi ustrezen zahtevek, tako da pripravi niz v obliki, kot je definiran v poglavju 4.7.3 (Ponastavitev parametrov delovanja), in ga pošlje na WLCService. Za komunikacijo uporabimo že obstoječi in pripravljen remoting kanal.

Poglavje 8 Sklepne ugotovitve

V diplomskem delu sem poskušal čim bolj celovito rešiti povsem praktično in vsakdanjo težavo – povezati različne tehnologije z namenom, da se ustvari preprost, a uporaben produkt. Kot rezultat je nastala domača rešitev za daljinsko spremljanje nivoja vode v zbiralniku.

Če povzamemo, je diplomsko delo v grobem zajemalo tri glavne segmente: razvoj strojne opreme oziroma merilnika, strežnik in mobilno spletno aplikacijo. Pri strojni opremi se je za čudovito izhodišče vsekakor izkazala odprtokodna platforma Arduino. Pri tem še zdaleč niso bile izčrpane vse možnosti.

Med razvojem in testiranjem sem kot zanimivost spremljal tudi stanje temperature v zbiralniku. Predvsem pa sem se ukvarjal z načinom avtonomnega delovanja strojne opreme in zmanjševanjem porabe energije. Izkazalo se je, da je največji porabnik napetostni regulator na plošči Arduino. Skupna poraba toka je približno 45 mA, kar bi pomenilo, da Arduino na klasičnem baterijskem napajanju 9 V deluje slab dan. Če damo čip v način spanja, lahko porabo zmanjšamo na približno 35 mA, kar pa je še vedno preveč.

Če čip ATmega328 uporabimo neodvisno, na primer na protoboard plošči, se poraba drastično zmanjša na nekaj mA. Če čip kasneje damo v način spanja, dobimo že zelo dobre rezultate, saj poraba pade na 1 uA ali manj. Avtonomno delovanje na omenjeni bateriji bi tako bilo zagotovljeno za vsaj leto dni. Najboljše rezultate sem dobil tako, da sem uporabil zunanji čip RTC (angl. *Real Time Clock*), ki skrbi za bujenje mikrokrmilnika, sam pa ima zanemarljivo porabo (DS1337). S tem dobimo na mikrokrmilniku tudi realni čas, DS1337 namreč vodi uro in datum. Seveda bi bilo tukaj v nadaljevanju potrebno poskrbeti tudi za varčevanje z energijo na modulu XBee.

Tako kot na strojni opremi je tudi na aplikativnem delu, pri uporabniški aplikaciji še ogromno možnosti za izboljšave. Ena pot, ki sem jo med projektom tudi delno preizkusil, je bila izgradnja aplikacije Android kot odjemalca. Aplikacija Android je tako komunicirala direktno s servisom WLC preko vtičnic (angl. *sockets*). Prednost je proženje opozoril ali obvestil neposredno na mobilni napravi brez uporabniške interakcije (uporabimo podoben model servisov kot na platformi Windows).

Za končni izdelek sem se odločil, da uporabim mobilno spletno aplikacijo. Tovrstne spletne aplikacije so vse bolj popularne in se tudi rahlo približujejo pravim (angl. *native*) aplikacijam. Prednost je razvoj na običajno dobro poznanih spletnih tehnologijah in seveda neodvisnost od mobilne platforme.

Končni izdelek pa ni le rešitev merjenja nivoja vode, temveč je lahko osnova za različne rešitve s področja telemetrije. To lahko dosežemo s priklopom novih tipal, posodobitvijo programa na mikrokrmilniku, razširitvijo našega komunikacijskega protokola in ne nazadnje s posodobitvijo končnega uporabniškega vmesnika.

Literatura

[1] Arduino [Online]. Dosegljivo: https://www.arduino.cc/. [Dostopano 12.7.2016].

[2] A. Štern, J. Guna. Arduino kot telematska platforma v pedagoškem procesu [Online]. Dosegljivo: http://www.ltfe.org/wp-content/uploads/2013/09/sternarduinop.pdf. [Dostopano 12.7.2016].

[3] Arduino compare board specs [Online]. Dosegljivo: https://www.arduino.cc/en/Products/Compare. [Dostopano 12.7.2016].

[4] Atmel Atmega328P Datasheet [Online]. Dosegljivo: http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_complete.pdf. [Dostopano 12.7.2016].

[5] Spletna trgovina Sparkfun [Online]. Dosegljivo: https://www.sparkfun.com/. [Dostopano 12.7.2016].

[6] Senzor razdalje Sharp GP2D12 datasheet [Online]. Dosegljivo: http://www.trossenrobotics.com/productdocs/GP2D12.pdf. [Dostopano 12.7.2016].

[7] Adafruit Xbee adapter [Online]. Dosegljivo: https://learn.adafruit.com/xbee-radios. [Dostopano 12.7.2016].

[8] XBee Pro Datasheet [Online]. Dosegljivo: https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf. [Dostopano 12.7.2016].

[9] .NET Remoting [Online]. Dosegljivo: https://msdn.microsoft.com/en-us/library/kwdt6w2k%28v=vs.71%29.ASPX. [Dostopano 12.7.2016].

[10] Ext.NET [Online]. Dosegljivo: http://ext.net/. [Dostopano 12.7.2016].

[11] Sencha ExtJS [Online]. Dosegljivo: https://www.sencha.com. [Dostopano 12.7.2016].